

Logix5000 Controllers Design Considerations

Catalog Numbers 1756 ControlLogix, 1756 GuardLogix, 1768 CompactLogix, 1768 Compact GuardLogix, 1769 CompactLogix, 1789 SoftLogix, CompactLogix 5370



Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.



WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.



ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

IMPORTANT

Identifies information that is critical for successful application and understanding of the product.

Labels may also be on or inside the equipment to provide specific precautions.



SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.



BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.



ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Allen-Bradley, CompactBlock Guard I/O, CompactLogix, ControlFLASH, ControlLogix, DH+, FactoryTalk, FLEX, GuardLogix, Kinetix, Logix5000, MicroLogix, PanelBuilder, PanelView, PhaseManager, PLC-2, PLC-3, PLC-5, POINT I/O, POINT Guard I/O, Rockwell Automation, Rockwell Software, RSBizWare, RSFieldbus, RSLinx, RSLinx 5000, RSNetWorx, RSView, SLC, SoftLogix, Stratix, Stratix 2000, Stratix 5700, Stratix 6000, Stratix 8000, Stratix 8300, Studio 5000, Studio 5000 Logix Designer, SynchLink, and Ultra are trademarks of Rockwell Automation, Inc.

ControlNet, DeviceNet, and EtherNet/IP are trademarks of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Logix5000 Controller Comparison - CompactLogix, ControlLogix, GuardLogix

Table 1 - CompactLogix, ControlLogix, and GuardLogix Characteristics

Characteristic	CompactLogix™ 1769-L30ER, 1769-L30ER-NSE, 1769-L30ERM, 1769-L33ER, 1769-L33ERM, 1769-L36ERM	CompactLogix 1769-L24ER-BB1B, 1769-L24ER-QBFC1B, 1769-L27ERM-QBFC1B	CompactLogix 1769-L16ER-BB1B, 1769-L18ER-BB1B, 1769-L18ERM-BB1B	ControlLogix® 1756-L71, 1756-L72, 1756-L73, 1756-L74, 1756-L75, 1756-L73XT GuardLogix® 1756-L72S, 1756-L73S, 1756-L73SXT
Controller tasks: • Continuous • Periodic • Event	32 • 100 programs/task, Version 23 and earlier • 1000 programs/task, max: Version 24 or later	32 • 100 programs/task, Version 23 and earlier • 1000 programs/task, max: Version 24 or later	32 • 100 programs/task, Version 23 and earlier • 1000 programs/task, max: Version 24 or later	32 • 100 programs/task, Version 23 and earlier • 1000 programs/task, max: Version 24 or later
Event tasks	Consumed tag, EVENT instruction, axis, and motion event triggers	Consumed tag, EVENT instruction, axis, and m25-7.50otion event triggers	Consumed tag, EVENT instruction, axis, and motion event triggers; this controller also executes an Event task from its embedded I/O modules	All event triggers
User memory	• 1769-L30ER, 1769-L30ER-NSE, 1769-L30ERM: 1MB • 1769-L33ER, 1769-L33ERM: 2 MB • 1769-L36ERM: 3 MB	• 1769-L24ER: 750 KB • 1769-L27ERM: 1 MB	• 1769-L16ER: 384 KB • 1769-L18ER, 1769-L18ERM: 512 KB	• 1756-L71: 2 MB • 1756-L72: 4 MB • 1756-L72S, 1756-L73SXT: 4 MB + 2 MB safety • 1756-L73, 1756-L73XT: 8 MB • 1756-L73S: 8 MB + 4 MB safety • 1756-L74: 16 MB • 1756-L75: 32 MB
Memory card	Secure Digital	Secure Digital	Secure Digital	Secure Digital
Built-in ports	2 EtherNet/IP and 1 USB	2 EtherNet/IP and 1 USB	2 EtherNet/IP and 1 USB	1 USB
Communication options	• Dual EtherNet/IP ports • DeviceNet	• Dual EtherNet/IP ports • DeviceNet	• Dual EtherNet/IP ports	• EtherNet/IP (standard and safety) • ControlNet (standard and safety) • DeviceNet (standard and safety) • DH+™ • Remote I/O • SynchLink™
Controller connections	256	256	256	500
Network connections	• 1769-L30ER, 1769-L30ER-NSE, 1769-L30ERM: 16 • 1769-L33ER, 1769-L33ERM: 32 • 1769-L36ERM: 48	• 1769-L24ER-BB1B, 1769-L24ER-QBFC1B, 8 • 1769-L27ERM-QBFC1B, 16	• 1769-L16ER-BB1B: 4 • 1769-L18ER-BB1B, 1769-L18ERM-BB1B: 8	Per module: • 128 ControlNet (CN2/B) • 40 ControlNet (CNB) • 256 EtherNet/IP; 128 TCP (EN2x) • 128 EtherNet/IP; 64 TCP (ENBT)
Controller redundancy	—	—	—	Supported with restrictions
Integrated motion	1769-L30ERM, 1769-L33ERM, and 1769-L36ERM support integrated motion on an EtherNet/IP network	1769-L27ERM-QBFC1B supports integrated motion on an EtherNet/IP network	1769-L18ERM-BB1B supports integrated motion on an EtherNet/IP network	Integrated motion on an EtherNet/IP network SERCOS interface Analog options
Programming languages	• Relay ladder • Structured text • Function block • SFC	• Relay ladder • Structured text • Function block • SFC	• Relay ladder • Structured text • Function block • SFC	• Standard task: all languages • Safety task: relay ladder, safety application instructions

Notes:

This manual contains new and updated information. Changes throughout this revision are marked by change bars, as shown to the right of this paragraph.

Topic	Page
Logix5000™ Controller Comparison Table: <ul style="list-style-type: none">Removed 1756-L6x, 1769-L23x, 1769-L3x, and 1769-L4x because they are no longer supported.Removed SoftLogix™ 5800 because the controller does not support the features in Studio 5000 Logix Designer Version® 24, but the controller does support Emulator in Version 24.Changed 1756-L72SXT to 1756-L73SXT.	3
Description of Access the Module Object Feature	33
Guidelines for Program Parameters	51
Comparison of Program Parameters and Add-On Instructions	52
Network Address Translation (NAT)	89

Notes:

Preface	Additional Resources	11
	Websites	11
	Chapter 1	
Logix5000 Controller Resources	Estimate Memory Use	16
	RSLinx Software Use of Logix5000 Controller Memory	16
	Compare PLC/SLC MEMORY	17
	Controller Connections	17
	Determine Total Connection Requirements	18
	CIP Sync	20
	Controller Mode	21
	Chapter 2	
Logic Execution	Decide When to Use Tasks, Programs, and Routines	24
	Specify Task Priorities	25
	Manage User Tasks	26
	Considerations that Affect Task Execution	27
	Configure a Continuous Task	29
	Configure a Periodic Task	29
	Configure an Event Task	30
	Guidelines to Configure an Event Task	30
	Additional Considerations for Periodic and Event Tasks	30
	Select a System Overhead Percentage	31
	Manage the System Overhead Timeslice Percentage	32
	Access the Module Object	33
	Create the Add-On Instruction	33
	Develop Application Code in Routines	34
	Comparison of Programming Languages	34
	Programming Methods	35
	Inline Duplication	35
	Indexed Routine	35
	Buffered Routine	36
	Controller Prescan of Logic	36
	Add-On Instruction Prescan Logic	37
	Controller Postscan of SFC Logic	37
	Add-On Instruction Postscan Logic	37
	Timer Execution	38
	SFC Step Timer Execution	38
	Edit an SFC Online	39
	Chapter 3	
Modular Programming Techniques	Guidelines for Code Reuse	41
	Naming Conventions	42
	Parameter Name Prefixes	44

Guidelines to for Subroutines 45

Guidelines for User-defined Data Types 46

 Naming Conventions for User-Defined Data Types 46

 UDT Member Order 46

Guidelines for Add-On Instructions 48

 Add-On Instruction Design Concepts 49

 Naming Conventions for Add-On Instructions 49

 Comparison of Subroutines and Add-On Instructions 49

 Comparison of Partial Import/Export and Add-On Instructions . 50

Guidelines for Program Parameters 51

 Comparison of Program Parameters and Add-On Instructions ... 52

Chapter 4

Address Data

Guidelines for Data Types 54

Arrays 55

Guidelines for Arrays 56

Indirect Addresses of Arrays 56

Guidelines for Array Indexes 57

Guidelines for User-defined Structures 58

Select a Data Type for Bit Tags 59

Serial Bit Addresses 60

Guidelines for String Data Types 61

PLC-5/SLC 500 Access of Strings 61

Configure Tags 62

Guidelines for Base Tags 62

Create Alias Tags 63

Guidelines for Data Scope 64

Guidelines for Tag Names 64

Guidelines for Extended Tag Properties 65

Tag Descriptions 66

Protect Data Access Control at Tag Level 66

Chapter 5

Produced and Consumed Data

Guidelines for Produced and Consumed Tags 67

Guidelines to Specify an RPI Rate for
Produced and Consumed Tags 68

Guidelines to Manage Connections for
Produced and Consumed Tags 69

Configure an Event Task Based on a Consumed Tag 69

Compare Messages and Produced/Consumed Tags 70

Chapter 6

Communicate with I/O

Buffer I/O Data 71

Guidelines to Specify an RPI Rate for I/O Modules 72

Communication Formats for I/O Modules 73

 Electronic Keying 75

	More Information	75
	Guidelines to Manage I/O Connections	76
	Control 1771 I/O Modules	77
	Communicate with HART Devices	78
	Communicate with FOUNDATION Fieldbus Devices	79
	Create Tags for I/O Data	81
	Controller Ownership	82
	Runtime/Online Addition of Modules	83
	Add Modules at Runtime/Online	84
	Design Considerations for Runtime/Online Addition of Modules	85
	Chapter 7	
Determine the Appropriate Network	EtherNet/IP Network Topology	88
	Guidelines for EtherNet/IP Networks	89
	Guidelines for Switches in EtherNet/IP Systems	90
	Determine Whether Your System Operates Properly	90
	Stratix Industrial Switches	91
	ControlNet Network Topology	91
	Guidelines for ControlNet Networks	92
	Guidelines for Unscheduled ControlNet Networks	93
	Compare Scheduled and Unscheduled ControlNet Communication	94
	DeviceNet Network Topology	94
	Guidelines for DeviceNet Networks	95
	Chapter 8	
Communicate with Other Devices	Cache Messages	98
	Message Buffers	98
	Outgoing Unconnected Buffers	99
	Guidelines for Messages	100
	Guidelines to Manage Message Connections	100
	Guidelines for Block Transfer Messages	101
	Map Tags	101
	Chapter 9	
FactoryTalk Alarms and Events System	Guidelines for Logix-based Alarm Instructions	103
	Changes in Logic	104
	Configure Logix-based Alarm Instructions	105
	Multiple Language Versions of Alarm Messages	106
	Alarm Process	107
	Alarm Log	108
	Programmatically Access Alarm Information	108
	Shelve, Suppress, or Disable Alarms	109

	Chapter 10	
Optimize an Application for Use with HMI	HMI Implementation Option.....	111
	Compare FactoryTalk View Site Edition and RSView32 Software...	112
	Guidelines for FactoryTalk View Software.....	112
	How RSLinx Software Communicates with Logix5000 Controllers .	113
	Compare RSLinx Classic and RSLinx Enterprise Software	114
	Guidelines for RSLinx Software	114
	Guidelines to Configure Controller Tags.....	115
	Reference Controller Data from FactoryTalk View Software	115
	Chapter 11	
Develop Equipment Phases	Guidelines for Equipment Phases	117
	Equipment Phase Instructions.....	118
	Chapter 12	
Manage Firmware	Guidelines to Manage Controller Firmware	119
	Compare Firmware Options	120
	Guidelines for the Firmware Supervisor	120
	Access Firmware	122
Glossary	123
Index	129

Additional Resources

These documents contain additional information about Logix5000 controllers.

Resource	Description
<ul style="list-style-type: none"> EtherNet/IP Modules in Logix5000 Control Systems User Manual, ENET-UM001 ControlNet Modules in Logix5000 Control Systems User Manual, CNET-UM001 DeviceNet Modules in Logix5000 Control Systems User Manual, DNET-UM004 	Networks
<ul style="list-style-type: none"> Logix5000 Common Procedures Programming Manual, 1756-PM001 Logix5000 Controllers General Instructions Reference Manual, 1756-RM003 Logix5000 Controllers Process Control and Drives Instructions Reference Manual, 1756-RM006 PhaseManager User Manual, LOGIX-UM001 Logix5000 Controllers Motion Instructions Reference Manual, MOTION-RM002 Logix5000 Controllers Import/Export Reference Manual, 1756-RM084 	Logix5000 Controllers
<ul style="list-style-type: none"> ControlLogix System User Manual, 1756-UM001 Motion Configuration and Startup User Manual, MOTION-UM001 Motion Coordinate System User Manual, MOTION-UM002 	ControlLogix Controllers
<ul style="list-style-type: none"> CompactLogix 5370 Controllers User Manual, 1769-UM021 1768 CompactLogix System User Manual, 1768-UM001 1769 CompactLogix System User Manual, 1769-UM011 1769 Packaged CompactLogix Controllers Quick Start and User Manual, IASIMP-OS010 	CompactLogix Controllers
<ul style="list-style-type: none"> SoftLogix System User Manual, 1789-UM002 	SoftLogix Controllers

You can view or download publications at <http://www.rockwellautomation.com/literature/>. To order paper copies of technical documentation, contact your local Allen-Bradley distributor or Rockwell Automation sales representative.

Websites

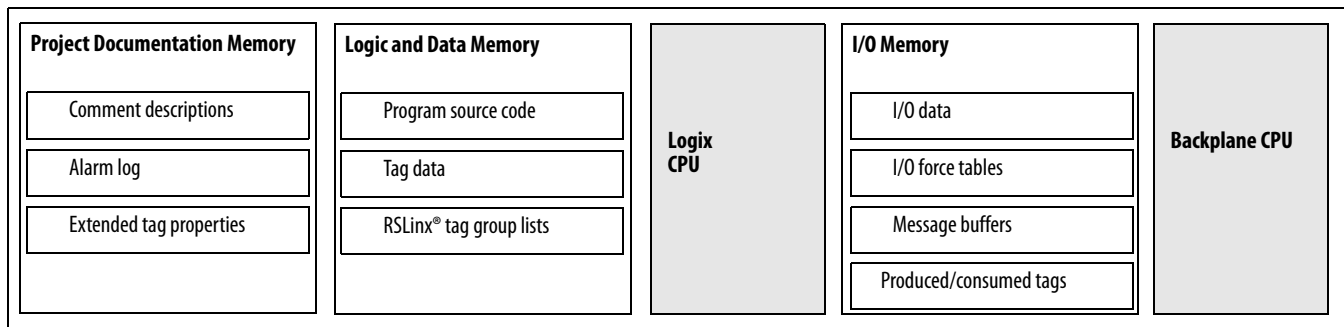
Resource	Description
http://www.ab.com/logix/	Logix Product Information
http://www.ab.com/networks/	Network Product Information
http://www.rockwellautomation.com/support/ In the left pane under Downloads, select Software Updates.	Software Updates (product serial number required)
Http://www.rockwellautomation.com/support In the left pane under Downloads, select Firmware Updates.	Firmware Updates (product serial number required)
http://www.ab.com/networks/eds/	Rockwell Automation® EDS Files
http://samplecode.rockwellautomation.com	Studio 5000® Sample Code

Notes:

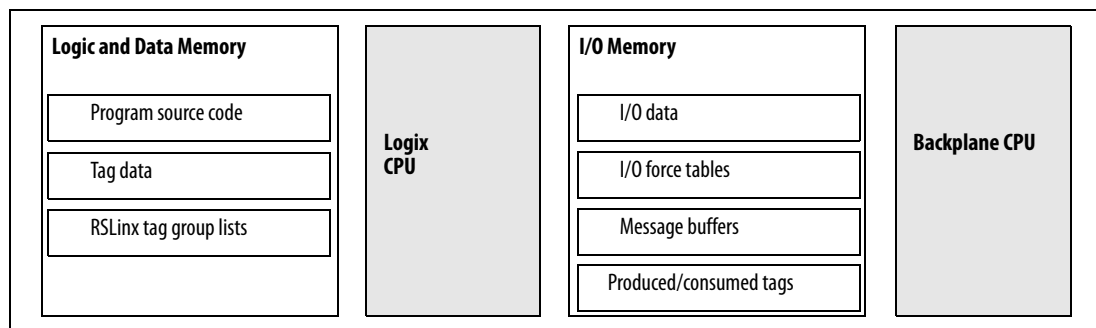
Logix5000 Controller Resources

Topic	Page
Estimate Memory Use	16
Controller Connections	17
Determine Total Connection Requirements	18
CIP Sync	20
Controller Mode	21

1756-L7x ControlLogix controllers - Memory is separated into isolated sections.



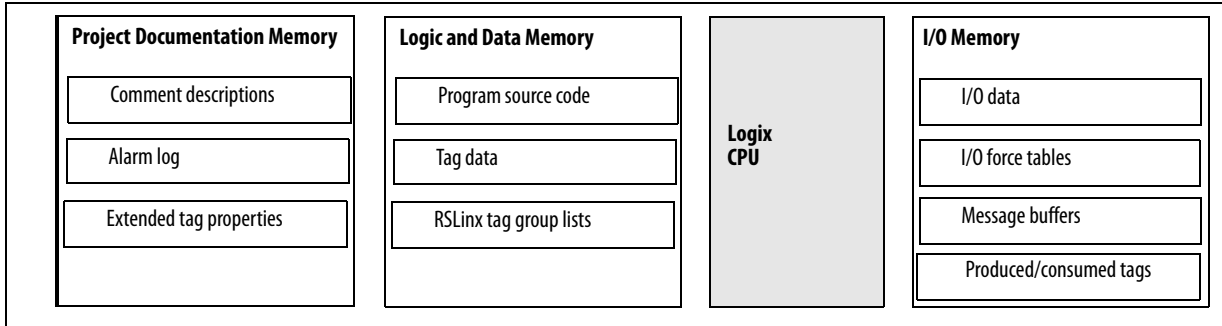
1768 CompactLogix and 1756-L6x ControlLogix controllers - Memory is separated into isolated sections.



The Logix CPU executes application code and messages. The backplane CPU transfers I/O memory and other module data on the backplane. This CPU operates independently from the Logix CPU, so it sends and receives I/O information asynchronous to program execution.

TIP CPU usage is based on the number of devices in the I/O tree. About 6% of the L7x CPU is used for every 100 devices in the I/O tree.

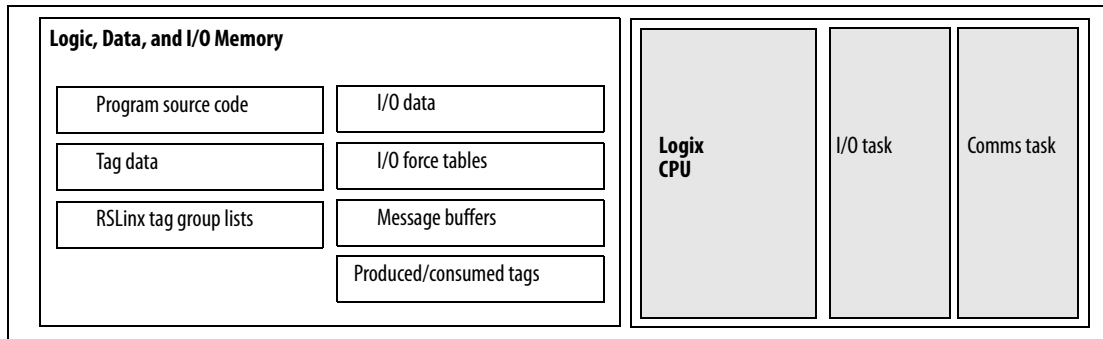
CompactLogix 5370 controllers - Memory is separated into isolated segments.



The Logix CPU executes application code and messages.

Controller	I/O Task Priority	Communication Task Priority
CompactLogix 5370	6	12

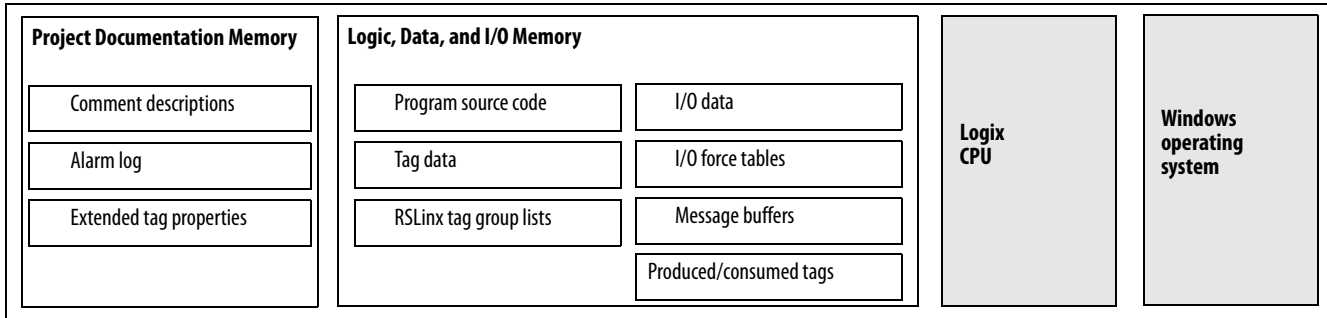
1769 CompactLogix controllers - Memory is in one, contiguous section.



These controllers have one CPU that performs all operations. Isolated tasks perform I/O and communication and interact with networks. These tasks simulate the backplane CPU.

Controller	I/O Task Priority	Communication Task Priority
1769 CompactLogix	6	12

SoftLogix controllers - Memory is in one, contiguous section.



The SoftLogix controller has one CPU that works with the Windows operating system to perform all operations. Rather than using controller priority levels for I/O and communication tasks, the SoftLogix controller uses Windows priority levels for these tasks.

Controller	I/O Task Priority	Communication Task Priority
SoftLogix	Windows priority 16 (Idle)	Windows priority 16 (Idle)

For all controllers, memory is used at runtime for the following:

- Message processing
- RSLinx data handling to store tag groups
- Online edits to store edit rungs
- Graphical trends to buffer data

Estimate Memory Use

The equations provide an estimate of the memory that is needed for a controller.

IMPORTANT If you configure controllers for redundancy, you must double the memory resources that are required for a non-redundant application.

Controller tasks	_____	* 4,000	=	_____	bytes (minimum 1 needed)
Digital I/O points	_____	* 400	=	_____	bytes
Analog I/O points	_____	* 2,600	=	_____	bytes
DeviceNet modules¹	_____	* 7,400	=	_____	bytes
Other communication modules²	_____	* 2,000	=	_____	bytes
Motion axis	_____	* 8,000	=	_____	bytes
FactoryTalk[®] alarm instruction	_____	* 2,200	=	_____	bytes (per alarm)
FactoryTalk subscriber	_____	* 2,000	=	_____	bytes (per subscriber)
		Total	=	_____	bytes

¹The first DeviceNet module is 7400 bytes. Additional DeviceNet modules are 5800 bytes each.

²Count all communication modules in the system, not just the modules in the local chassis. The count includes device connection modules, adapters, and ports on PanelView™ terminals.

IMPORTANT Reserve 20...30% of the controller memory to accommodate growth.

RSLinx Software Use of Logix5000 Controller Memory

The amount of memory that RSLinx software needs depends on the type of data RSLinx software reads. These equations provide a memory estimate.

RSLinx overhead (per connection)	_____	* 1345	=	_____	bytes (four connections by default)
Individual tags	_____	* 45	=	_____	bytes
Arrays / structures	_____	* 7	=	_____	bytes
		Total	=	_____	bytes

You can consolidate tags into an array or a structure to reduce the communication overhead and the number of connections that are used to obtain the data.

Compare PLC/SLC MEMORY

The Logix5000 controllers use compiled instructions to provide faster execution times than PLC or SLC™ processors. The compiled instructions use more memory when compared to the instructions in PLC and SLC processors.

If you have a PLC/SLC program, you can estimate the number of bytes it takes in a Logix5000 controller by the following equation:

$$\text{number PLC/SLC words} * 18 = \text{number of Logix5000 bytes}$$

Controller Connections

A Logix5000 controller uses a connection to establish a communication link between two devices. Connections can be made to the following:

- Controller to local I/O modules or local communication modules
- Controller to remote I/O or remote communication modules
- Controller to remote I/O (rack optimized) modules

For more information on connections for I/O, see [Communicate with I/O on page 71](#).

- Produced and consumed tags

For more information, see [Produced and Consumed Data on page 67](#).

- Messages

For more information, see [Communicate with I/O on page 71](#).

- Access to RSLogix 5000® software
- RSLinx software access for HMI or other software applications

The controllers have different communication limits.

Communication Attribute	1756-L7x ControlLogix	1756-L6x ControlLogix and SoftLogix	1769 CompactLogix	CompactLogix 5370	1768 CompactLogix
Connections	500	250	100	256	250
Cached messages ⁽¹⁾	32 for messages and block transfers combined				
Unconnected receive buffers	3				
Unconnected transmit buffers	Default 20 (can be increased to 40)		Default 10 (can be increased to 40)		

(1) See [Communicate with Other Devices on page 97](#) for more information about messages and buffers.

The limit of connections can ultimately reside in the communication module you use for the connection. If a message path routes through a communication module, the connection that is related to the message also counts toward the connection limit of that communication module.

Controller	Communication Device	Supported Connections
ControlLogix	1756-CN2R, 1756-CN2RXT	100 CIP connections (any combination of scheduled and message connections)
	1756-CN2/B	128 CIP connections
	1756-CNB, 1756 -CNBR	64 CIP connections depending on RPI, recommend that you use only 48 connections (any combination of scheduled and message connections)
	1756-EN2F, 1756-EN2T, 1756-EN2TR, 1756-EN2TXT, 1756-EN3TR	256 CIP connections 128 TCP/IP connections
	1756-ENBT 1756-EWEB	128 CIP connections 64 TCP/IP connections
1768 CompactLogix	1768-ENBT 1768-EWEB	64 CIP connections 32 TCP/IP connections
1769 CompactLogix	1769-L32C, 1769-L35CR	32 CIP connections depending on RPI, as many as 22 connections can be scheduled The remaining connections (or all 32, if you have no scheduled connections) can be used for message connections
	1769-L32E, 1769-L35E	32 CIP connections 64 TCP/IP connections
	1769-L23Ex	32 CIP connections 12 TCP/IP connections
CompactLogix 5370	Built-in Ethernet ports	See the CompactLogix 5370 Controllers User Manual, publication 1769-UM021 , for information on how to count EtherNet/IP nodes on the I/O Configuration section of RSLogix 5000 software.
SoftLogix 5800	1784-PCICS	128 CIP connections 127 can be scheduled connections

Determine Total Connection Requirements

The total connections for a Logix5000 controller include both local and remote connections. Counting local connections is not an issue for CompactLogix controllers. They support the maximum number of modules that are permitted in their systems.

When designing your CompactLogix 5370 controllers, you must consider these resources:

- EtherNet/IP network nodes
- Controller connections

For more information, see the CompactLogix 5370 Controllers User Manual, publication [1769-UM021](#).

The ControlLogix and SoftLogix controllers support more communication modules than the other controllers, so you must tally local connections to make sure that you stay within the connection limit.

Use this table to tally **local** connections.

Connection Type	Device Quantity	x	Connections per Module	=	Total Connections
Local I/O module (always a direct connection)		x	1	=	
SERCOS Motion module		x	3	=	
ControlNet communication module		x	0	=	
EtherNet/IP communication module		x	0	=	
DeviceNet communication module		x	2	=	
DH+ /Remote I/O communication module		x	1	=	
DH-485 communication module		x	1	=	
RSLogix 5000 software access to controller		x	1	=	
Total					

IMPORTANT A redundant system uses eight connections in the controller.

The communication modules that you select determine how many remote connections are available. Use this table to tally **remote** connections.

Connection Type	Device Quantity	x	Connections per Module	=	Total Connections
Remote ControlNet communication module Configured as a direct (none) connection Configured as a rack-optimized connection		x	0 or 1	=	
Remote EtherNet/IP communication module Configured as a direct (none) connection Configured as a rack-optimized connection		x	0 or 1	=	
Remote device over a DeviceNet network (accounted for in rack-optimized connection for local DeviceNet module)		x	0	=	
Safety device on a DeviceNet or EtherNet/IP network		x	2	=	
Other remote communication adapter		x	1	=	
Distributed I/O module (individually configured for a direct connection)		x	1	=	
Produced tag and first consumer Each additional consumer		x	2 1	=	
Consumed tag		x	1	=	
Connected message (CIP Data Table Read/Write and DH+)		x	1	=	
Block transfer message		x	1	=	
RSLinx software access for HMI or other software applications		x	4	=	
RSLinx® Enterprise software for HMI or other software applications		x	5	=	
Total					

CIP Sync

CIP Sync is a time synchronization implementation that incorporates IEEE-1588 standards on the EtherNet/IP protocol. CIP Sync provides the control system access to synchronization information and transport and routing of a system clock on standard CIP networks.

CIP Sync offers the following features:

- Precision Time Protocol (PTP)
- Nanosecond resolution +/- 100 nanosecond synchronization (hardware assist clock)
- Master clock reference
- No longer need application code or software to synchronize clocks between controller, HMI, and other control hardware.
- Open standard lets compatibility with most IEEE-1588 v2 products exist, letting the integration with GPS and other IT layer devices occur.
- Alarm system automatically picks up time stamps from CIP Sync system time
- System self-heals, so that if one clock master fails the rest arbitrate to find the next best clock master.

A ControlLogix controller or 1756-EN2T can become a system clock master. Other Logix5000 controllers can require application code.

The controller or networked device that wins system clock arbitration is the Grandmaster clock. The wall clock time can only be set from the system Grand Master device. If you adjust a controller clock, the controller could reject that time if it is not or does not become the Grandmaster clock.

You can configure the system clock via RSLogix 5000 software, version 18 and later, and programmatically via GSV/SSV instructions. Use a GSV/SSV instruction with the Time Sync object to do the following:

- Enable or disable CIP Sync
- Get or set the time
- Set priority to override other masters
- Get synchronization status
- Get current PTP master status and state information

Controller Mode

The controller mode switch provides a mechanical means to enhance controller and control system security. You must physically move the switch to change the operating mode from RUN to REM or to PROG.

Remote lets you change the operational mode to REM RUN or REM PROG via RSLogix 5000 software.

IMPORTANT During runtime, we recommend that you place the controller mode switch in RUN mode and remove the key (if applicable) from the switch. This practice helps discourage unauthorized access to the controller or potential tampering with the controller program, configuration, or device firmware. Place the mode switch in REM or PROG mode during controller commissioning, maintenance, and whenever temporary access is necessary to change the program, configuration, or firmware.

For more information on controller mode switches, see the ControlLogix System User Manual, publication [1756-UM001](#).

Notes:

Logic Execution

Topic	Page
Decide When to Use Tasks, Programs, and Routines	24
Specify Task Priorities	25
Manage User Tasks	26
Considerations that Affect Task Execution	27
Configure a Continuous Task	29
Configure a Periodic Task	29
Configure an Event Task	30
Select a System Overhead Percentage	31
Manage the System Overhead Timeslice Percentage	32
Access the Module Object	33
Develop Application Code in Routines	34
Programming Methods	35
Controller Prescan of Logic	36
Controller Postscan of SFC Logic	37
Timer Execution	38
Edit an SFC Online	39

The controller operating system is a ct2000LAK pre-emptive multitasking system that is IEC 61131-3 compliant.

Tasks to configure controller execution

A task provides scheduling and priority information for a set of one or more programs. You can configure tasks as either continuous, periodic, or event.

Programs to group data and logic

A task contains programs, each with its own routines and program-scoped tags. Once a task is triggered (activated), the programs that are assigned to the task execute in the order in which they are listed in the Controller Organizer.

Programs are useful for projects that multiple programmers develop. During development, the code in one program that uses program-scoped tags can be duplicated in a second program to minimize the possibility of tag names colliding.

With firmware revision 15, tasks can contain programs and equipment phases.

Routines to encapsulate executable code that is written in one programming language

Routines contain the executable code. Each program has a main routine that is the first routine to execute within a program. Use logic, such as the Jump to Subroutine (JSR) instruction, to call other routines. You can also specify an optional program fault routine.

See [Develop Application Code in Routines on page 34](#) for information on selecting programming languages, and how the controller prescans and postscans logic.

Decide When to Use Tasks, Programs, and Routines

Use these considerations to determine when to use a task, program, or routine.

Comparison	Task	Program and Equipment Phase	Routine
Quantity available	Varies by controller (4, 6, 8, or 32)	32 program and equipment phases (combined) per task (100 for ControlLogix and SoftLogix controllers)	Unlimited number of routines per program
Function	Determines how and when code is executed	Organizes groups of routines that share a common data area and function.	Contains executable code (relay ladder, function block diagram, sequential function chart, or structured text)
Use	<ul style="list-style-type: none"> • Most code is expected to reside in a continuous task • Use a periodic task for slower processes or when time-based operation is critical • Use an event task for operations that require synchronization to a specific event 	<ul style="list-style-type: none"> • Put major equipment pieces or plant cells into isolated programs • Use programs to isolate different programmers or create reusable code • Configurable execution order within a task • Isolate individual batch phases or discrete machine operations 	<ul style="list-style-type: none"> • Isolate machine or cell functions in a routine • Use the appropriate language for the process • Modularize code into subroutines that can be called multiple times
Considerations	<ul style="list-style-type: none"> • A high number of tasks can be difficult to debug • Can disable output processing on some tasks to improve performance • Tasks can be inhibited to prevent execution • Do not configure multiple tasks at the same priority 	<ul style="list-style-type: none"> • Data spanning multiple programs must go into controller-scoped area • Listed in the Controller Organizer in the order of execution 	<ul style="list-style-type: none"> • Subroutines with multiple calls can be difficult to debug • Data can be referenced from program-scoped and controller-scoped areas • Calling many routines impacts scan time • Listed in the Controller Organizer as Main, Fault, and then alphabetically

For more information about equipment phases, see [Develop Equipment Phases on page 117](#).

Specify Task Priorities

Each task in the controller has a priority level. A higher priority task (such as 1) interrupts any lower priority task (such as 15). The continuous task has the lowest priority; periodic or event tasks always interrupt continuous tasks.

Logix5000 Controller	User Tasks Supported	Available Priority Levels
ControlLogix	32	15
CompactLogix 5370	32	15
1768-L43, 1769-L45 CompactLogix	16	15
1769-L35CR, 1769-L35E CompactLogix	8	15
1769-L32C, 1769-L32E CompactLogix	6	15
1769-L31 CompactLogix	4	15
1769-L23E-QB1B, 1769-L23E-QBFC1B, 1769-L23-QBFC1B CompactLogix	3	15
SoftLogix 5800	32	3

The Logix5000 controller has these types of tasks.

Priority	User Task	Description
Highest	N/A	CPU overhead - serial port and general CPU operations
	N/A	Motion planner - executed at coarse update rate
	N/A	Safety task - safety logic
	N/A	Redundancy task - communication in redundant systems
	N/A	Trend data collection - high-speed collection of trend data values
	Priority 1 Event/Periodic	User defined
	Priority 2 Event/Periodic	User defined
	Priority 3 Event/Periodic	User defined
	Priority 4 Event/Periodic	User defined
	Priority 5 Event/Periodic	User defined
	Priority 6 Event/Periodic	User defined 1769 CompactLogix controllers process I/O as a periodic task based on the chassis RPI setting
	Priority 7 Event/Periodic	User defined
	Priority 8 Event/Periodic	User defined
	Priority 9 Event/Periodic	User defined
	Priority 10 Event/Periodic	User defined
Lowest	Priority 11 Event/Periodic	User defined
	Priority 12 Event/Periodic	User defined CompactLogix communication and scheduled connection maintenance
	Priority 13 Event/Periodic	User defined
	Priority 14 Event/Periodic	User defined
	Priority 15 Event/Periodic	User defined
	Continuous	Message handler - based on system overhead timeslice

If a periodic or event task is executing when another is triggered, and both tasks are at the same priority level, the tasks' timeslice executes in 1 ms increments until one of the tasks completes execution.

Manage User Tasks

You can configure these user tasks.

If you want logic to execute	Use this task	Description
All of the time	Continuous task	The continuous task runs in the background. Any CPU time that is not allocated to other operations or tasks is used to execute the continuous task. <ul style="list-style-type: none"> The continuous task runs all of the time. When the continuous task completes a full scan, it restarts immediately. A project does not require a continuous task. If used, there can be only one continuous task.
<ul style="list-style-type: none"> At a constant period (such as every 100 ms) Multiple times within the scan of your other logic 	Periodic task	A periodic task performs a function at a specific time interval. Whenever the time for the periodic task expires, the periodic task: <ul style="list-style-type: none"> Interrupts any lower priority tasks. Executes one time. Returns control to where the previous task left off.
Immediately when an event occurs	Event task	An event task performs a function only when a specific event (trigger) occurs. Whenever the trigger for the event task occurs, the event task: <ul style="list-style-type: none"> Interrupts any lower priority tasks. Executes one time. Returns control to where the previous task left off. See Configure an Event Task on page 30 for the triggers for an event task. Some Logix5000 controllers do not support all triggers.

The user tasks that you create appear in the Tasks folder of the controller. The predefined system tasks do not appear in the Tasks folder and they do not count toward the task limit of the controller:

- Motion planner
- I/O processing
- System overhead
- Output processing

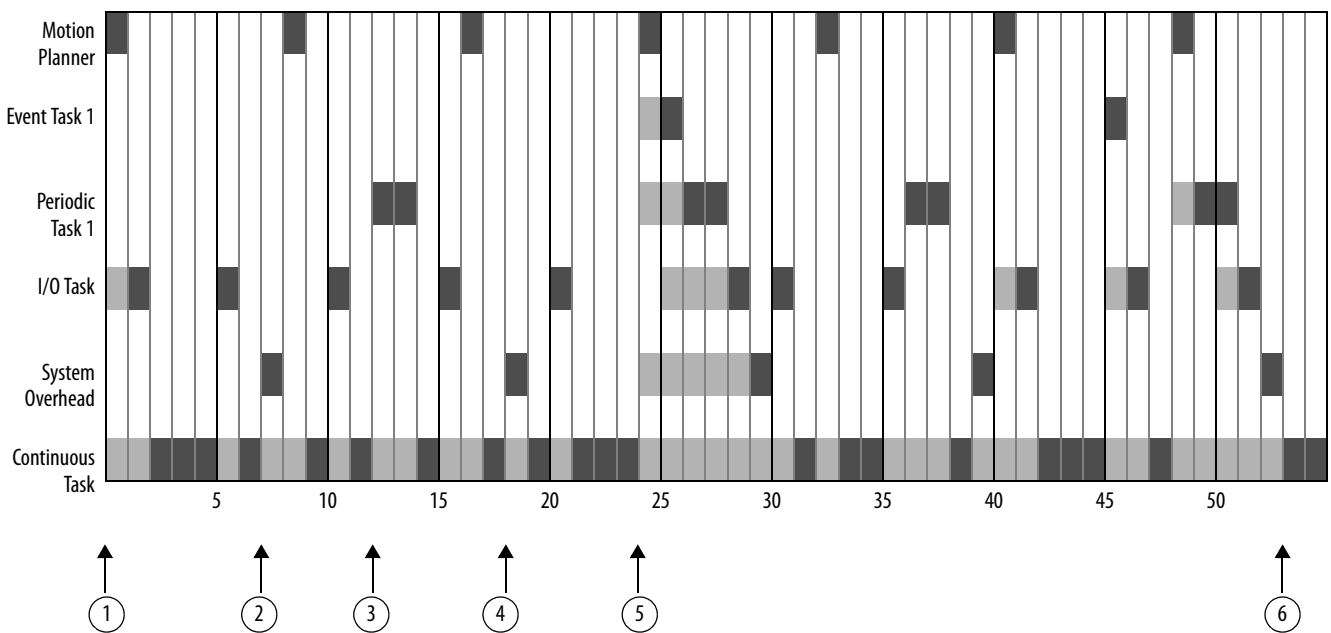
Considerations that Affect Task Execution

Consideration	Description
Motion planner	<p>The motion planner interrupts all other tasks, regardless of their priority.</p> <ul style="list-style-type: none"> • The number of axes and coarse update period for the motion group affect how long and how often the motion planner executes. • If the motion planner is executing when a task is triggered, the task waits until the motion planner is done. • If the coarse update period occurs while a task is executing, the task pauses to let the motion planner execute.
I/O processing	<p>CompactLogix and SoftLogix controllers use a dedicated periodic task to process I/O data. This I/O task:</p> <ul style="list-style-type: none"> • CompactLogix controllers, operates at priority 6. • SoftLogix controllers, operates at Windows priority 16 (Idle). • Higher-priority tasks take precedence over the I/O task and can affect processing. • Executes at the fastest RPI you have scheduled for the system. • Executes for as long as it takes to scan the configured I/O modules.
<p>System overhead</p> <p>See also Select a System Overhead Percentage on page 31.</p>	<p>System overhead is the time that the controller spends on message communication and background tasks.</p> <ul style="list-style-type: none"> • Message communication is any communication that you do not configure through the I/O configuration folder of the project, such as MSG instructions. • Message communication occurs only when a periodic or event task is not running. If you use multiple tasks, make sure that their scan times and execution intervals leave enough time for message communication. • System overhead interrupts only the continuous task. • The system overhead timeslice specifies the percentage of time (excluding the time for periodic or event tasks) that the controller devotes to message communication. • The controller performs message communication for up to 1 ms at a time and then resumes the continuous task. • Adjust the update rates of the tasks as needed to get the best trade-off between executing your logic and servicing message communication.
Output processing	<p>At the end of a task, the controller performs output processing for the output modules in your system. This processing depends on the number of output connections that are configured in the I/O tree.</p>
Too many tasks	<p>If you have too many tasks, then the following can occur:</p> <ul style="list-style-type: none"> • Continuous task can take too long to complete. • Other tasks can experience overlaps. If a task is interrupted too frequently or too long, it must be triggered again to complete its execution. • Controller communication can be slower. • If your application is designed for data collection, try to avoid multiple tasks.

This example depicts the execution of a project with these tasks.

Task	Priority	Period	Execution Time	Duration
Motion planner	N/A	8 ms (course update rate)	1 ms	1 ms
Event task 1	1	N/A	1 ms	1...2 ms
Periodic task 1	2	12 ms	2 ms	2...4 ms
I/O task—N/A to ControlLogix and SoftLogix controllers	7	5 ms (fastest RPI)	1 ms	1...5 ms
System overhead	N/A	Timeslice = 20%	1 ms	1...6 ms
Continuous task	N/A	N/A	20 ms	48 ms

Legend: Task executes. Task is interrupted (suspended).



Description	
①	Initially, the controller executes the motion planner and the I/O task (if one exists).
②	After executing the continuous task for 4 ms, the controller triggers the system overhead.
③	The period for periodic task 1 expires (12 ms), so the task interrupts the continuous task.
④	After executing the continuous task again for 4 ms, the controller triggers the system overhead.
⑤	The triggers occur for event task 1. Event task 1 waits until the motion planner is done. Lower priority tasks experience longer delays.
⑥	The continuous task automatically restarts.

Configure a Continuous Task

The continuous task is created automatically when you open an RSLogix 5000® software project. A continuous task is similar to how logic executes on PLC-5® and SLC™ 500 processors. A Logix5000 controller supports one continuous task, but a continuous task is not required. You can configure whether the task updates output modules at the end of the continuous task. You can change the continuous task to either a periodic or event task.

The CPU timeslices between the continuous task and system overhead. Each task switch between user task and system overhead takes more CPU time to load and restore task information.

RSLogix 5000 software, version 16 and later, forces at least 1 ms of execution time for the continuous task, regardless of the system overhead timeslice. This more efficiently uses system resources because letting shorter execution times of the continuous task exist means switching tasks more frequently.

System Overhead Timeslice %	Communication Execution (msec)	Continuous Task Execution (msec)
10	1	9
20	1	4
33	1	2
50	1	1
66	2	1
80	4	1
90	9	1

Configure a Periodic Task

A periodic task executes automatically based on a preconfigured interval. This task is similar to selectable timed interrupts in PLC-5® and SLC 500 processors. You can configure whether the task updates output modules at the end of the periodic task. After the task executes, it does not execute again until the configured time interval has elapsed.

If your application has a lot of communication, such as RSLinx communication, use a periodic task rather than a continuous task.

Configure an Event Task

An event task executes automatically based on a trigger event occurring or if a trigger event does not occur in a specific time interval. You configure whether the task updates output modules at the end of the task. After the task executes, it does not execute again until the event occurs again. Each event task requires a specific trigger.

Trigger	Description
Module Input Data State Change	With Logix5000 controllers, a remote input module (digital or analog) triggers an event task that is based on the change of state (COS) configuration for the module. Enable COS for only one point on the module. If you enable COS for multiple points, a task overlap of the event task can occur. <ul style="list-style-type: none"> The ControlLogix sequence of events modules (1756-IB16ISOE, 1756-IH16ISOE) use the Enable CST Capture feature instead of COS. The embedded input points on the 1769-L16ER-BB1B, 1769-L18ER-BB1B, and 1769-L18ERM-BB1B modules can be configured to trigger an event task when a COS occurs.
Consumed Tag	Only one consumed tag can trigger a specific event task. Use an IOT instruction in the producing controller to signal the production of new data.
Axis Registration 1 or 2	A registration input triggers the event task.
Axis Watch	A watch position triggers the event task.
Motion Group Execution	The coarse update period for the motion group triggers the execution of both the motion planner and the event task. Because the motion planner interrupts all other tasks, it executes first.
EVENT Instruction	Multiple EVENT instructions can trigger the same task.

For more information on event tasks, see:

- Logix5000 Controllers Common Procedures Programming Manual, publication [1756-PM001](#)
- Using Event Tasks with Logix5000 Controllers, publication [LOGIX-WP003](#)

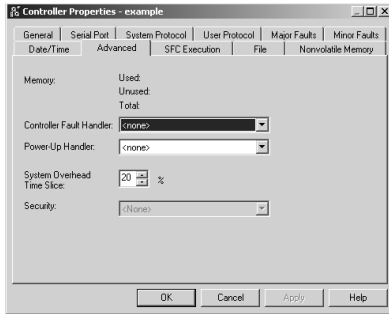
Guidelines to Configure an Event Task

Guideline	Description
Place the I/O module being used to trigger an event in the same chassis as the controller.	Placing the I/O module in a remote chassis adds more network communication and processing to the response time.
Limit events on digital inputs to one input bit on a module.	All inputs on a module trigger one event, so if you use multiple bits you increase the chance of a task overlap. Configure the module to detect change of state on the trigger input and turn off the other bits.
Set the priority of the event task as the highest priority on the controller.	If the priority of the event task is lower than a periodic task, the event task has to wait for the periodic task to complete execution.
Limit the number of event tasks.	Increasing the number of event tasks reduces the available CPU bandwidth and increases the chances of task overlap.

Additional Considerations for Periodic and Event Tasks

Consideration	Description
Amount of code in the event task	Each logic element (for example, rung, instruction, or structured text construct) adds to scan time.
Task priority	If the event task is not the highest priority task, a higher priority task can delay or interrupt the execution of the event task.
CPS and UID instructions	If one of these instructions are active, the event task cannot interrupt the currently executing task. (The task with the CPS or UID.)
Communication interrupts	Incoming character processing through the serial port interrupts a task, regardless of the priority of the task.
Motion planner	The motion planner takes precedence over event or periodic tasks
Trends	Trend data collection takes precedence over event or periodic tasks.
Output processing	You can disable output processing at the end of a task to reduce the amount of task processing time. As of RSLogix 5000 software, version 16, the Controller Organizer displays whether outputs processing is disabled.

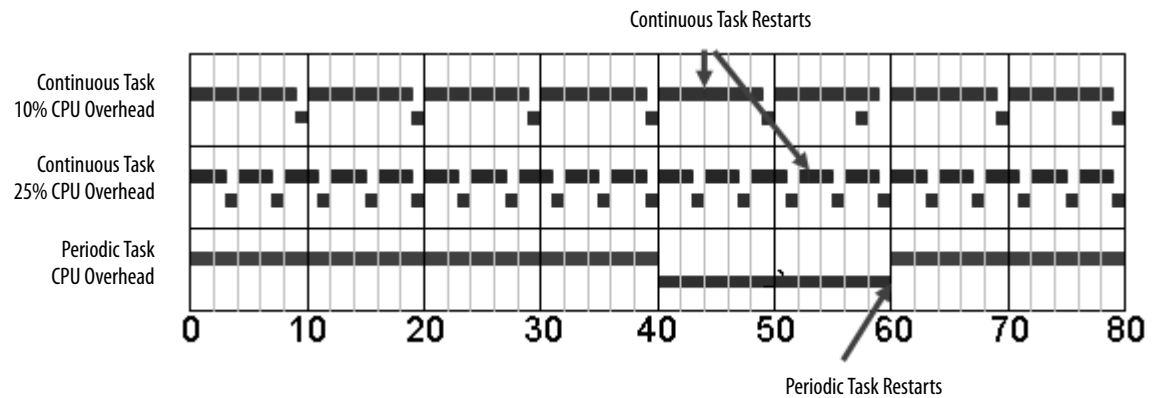
Select a System Overhead Percentage



The system overhead timeslice specifies the percentage of continuous task execution time that is devoted to communication and background redundancy functions. System overhead functions include the following:

- Communicating with programming and HMI devices (such as RSLogix 5000 software)
- Responding to messages
- Sending messages
- Serial port message and instruction processing
- Alarm instruction processing
- Redundancy qualification

The controller performs system overhead functions for up to 1 ms at a time. If the controller completes the overhead functions in less than 1 ms, it resumes the continuous task. The following chart compares a continuous and periodic task.



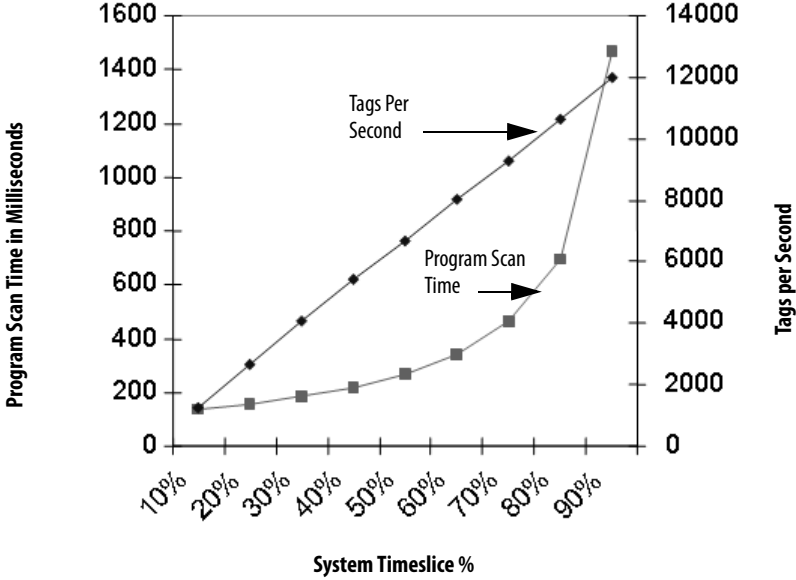
Example	Description
Continuous task 10% CPU overhead	In the top example, the system overhead timeslice is set to 10%. Given 40 ms of code to execute, the continuous task completes the execution in 44 ms. During a 60 ms period, the controller is able to spend 5 ms on communication processing.
Continuous task 25% CPU overhead	By increasing the system overhead timeslice to 25%, the controller completes the continuous task scan in 57 ms. The controller spends 15 ms of a 60 ms time span on communication processing.
Periodic task	Placing the same code in a periodic task yields even more time for communication processing. The bottom example assumes that the code is in a 60 ms periodic task. The code executes to completion and then goes dormant until the 60 ms, time-based trigger occurs. While the task is dormant, all CPU bandwidth can focus on communication. Because the code takes only 40 ms to execute, the controller can spend 20 ms on communication processing. Depending on the amount of communication to process during this 20 ms window, it can be delayed as it waits for other modules in the system to process all of the data that was communicated.

The Logix5000 CPU timeslices between the continuous task and system overhead. Each task switch between user task and system overhead takes additional CPU time to load and restore task information. You can calculate the continuous task interval as:

$$\text{ContinuousTime} = (100 / \text{SystemOverheadTimeSlice}\%) - 1$$

Manage the System Overhead Timeslice Percentage

As the system overhead timeslice percentage increases, time that is allocated to executing the continuous task decreases. If there is no communication for the controller to manage, the controller uses the communication time to execute the continuous task.

Consideration	Description																														
Continuous task always has at least 1 ms execution time	RSLogix 5000 software, version 16 and later, forces the continuous task to have at least 1 ms of execution time, regardless of the setting for the system overhead timeslice. This results in more efficient controller use because excessive swapping between tasks uses valuable CPU resources.																														
Impact on communication and scan time	<p>Increasing the system overhead timeslice percentage decreases execution time for the continuous task while it increases communication performance.</p> <p>Increasing the system overhead timeslice percentage also increases the amount of time it takes to execute a continuous task - increasing overall scan time.</p>  <p>The graph plots two metrics against System Timeslice % (10% to 90%). The left Y-axis is Program Scan Time in Milliseconds (0 to 1600), and the right Y-axis is Tags per Second (0 to 14000). The X-axis is System Timeslice % (10% to 90%). The 'Tags Per Second' series (diamonds) increases from approximately 1000 at 10% to 13000 at 90%. The 'Program Scan Time' series (squares) increases from approximately 100 ms at 10% to 1400 ms at 90%.</p> <table border="1"> <caption>Approximate data points from the graph</caption> <thead> <tr> <th>System Timeslice %</th> <th>Program Scan Time (ms)</th> <th>Tags per Second</th> </tr> </thead> <tbody> <tr><td>10%</td><td>100</td><td>1000</td></tr> <tr><td>20%</td><td>150</td><td>2000</td></tr> <tr><td>30%</td><td>200</td><td>3000</td></tr> <tr><td>40%</td><td>250</td><td>4000</td></tr> <tr><td>50%</td><td>300</td><td>5000</td></tr> <tr><td>60%</td><td>350</td><td>6000</td></tr> <tr><td>70%</td><td>400</td><td>7000</td></tr> <tr><td>80%</td><td>450</td><td>8000</td></tr> <tr><td>90%</td><td>1400</td><td>13000</td></tr> </tbody> </table>	System Timeslice %	Program Scan Time (ms)	Tags per Second	10%	100	1000	20%	150	2000	30%	200	3000	40%	250	4000	50%	300	5000	60%	350	6000	70%	400	7000	80%	450	8000	90%	1400	13000
System Timeslice %	Program Scan Time (ms)	Tags per Second																													
10%	100	1000																													
20%	150	2000																													
30%	200	3000																													
40%	250	4000																													
50%	300	5000																													
60%	350	6000																													
70%	400	7000																													
80%	450	8000																													
90%	1400	13000																													
Unused portion of system overhead timeslice	<p>With RSLogix 5000 software, version 16, you can configure any unused portion of the system overhead timeslice to:</p> <ul style="list-style-type: none"> Run the continuous task, which results in faster execution of application code and increases the variability of the program scan. Process communication, which results in more predictable and deterministic scan time for the continuous task. (This is for development and testing of an application to simulate communication.) 																														

Individual applications can differ, but the overall impact on communication and scan time remains the same. The data is based on a ControlLogix5555 controller running a continuous task with 5000 tags (no arrays or user-defined structures).

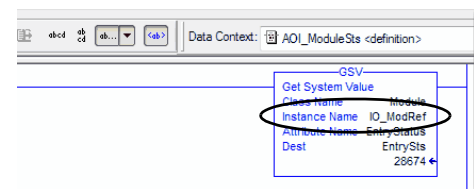
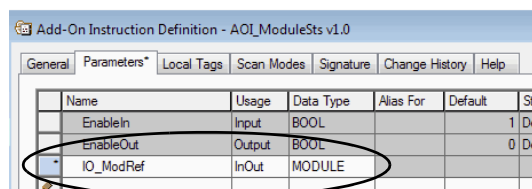
Access the Module Object

The MODULE object provides status information about a module. To select a particular module object, set the Object Name operand of the GSV/SSV instruction to the module name. The specified module must be present in the I/O Configuration section of the controller organizer and must have a device name.

Create the Add-On Instruction

With Logix Designer Application, version 24.00.00 and later, you can access the MODULE object directly from an Add-On Instruction. Previously, you could access the MODULE object data, but not from within an Add-On Instruction.

You must create a **Module Reference** parameter when you define the Add-On Instruction to access the MODULE object data. A Module Reference parameter is an InOut parameter of the MODULE data type that points to the MODULE Object of a hardware module. You can use module reference parameters in both Add-On Instruction logic and program logic.



For more information on the Module Reference parameter, see the Logix5000 Controllers Add On Instructions programming manual, publication [1756-PM010](#) and the Logix Designer application online help.

The MODULE object uses the following attributes to provide status information:

- EntryStatus
- FaultCode
- FaultInfo
- FWSupervisorStatus
- ForceStatus
- Instance
- LEDStatus
- Mode
- Path

The **Path** attribute is available with Logix Designer application, version 24.00.00 and later. This attribute provides a communication path to the module.

For more information on the attributes available in the MODULE object, see the Logix5000 Controllers General Instructions Reference Manual, publication [1756-RM003](#).

Develop Application Code in Routines

Each routine contains logic in one programming language. Choose a programming language that is based on the application.

Section of Code Represents	Language to Use
Continuous or parallel execution of multiple operations (not sequenced)	Relay ladder logic (LD)
Boolean or bit-based operations	
Complex logical operations	
Message and communication processing	
Machine interlocking	
Operations that service or maintenance personnel can interpret to troubleshoot the machine or process.	
Servo motion control	
Continuous process and drive control	Function block diagram (FBD)
Loop control	
Calculations in circuit flow	
High-level management of multiple operations	Sequential function chart (SFC)
Repetitive sequences of operations	
Batch process	
Motion control sequencing (via sequential function chart with embedded structure text)	
State machine operations	
Complex mathematical operations	Structured text (ST)
Specialized array or table loop processing	
ASCII string handling or protocol processing	

Comparison of Programming Languages

Comparison	Relay Ladder Logic	Function Block Diagram	Sequential Function Chart	Structured Text
Instruction categories	<ul style="list-style-type: none"> • Boolean • General and trig math • Timers and counters • Array management • Diagnostic • Serial port and messaging • ASCII manipulation • Motion control 	<ul style="list-style-type: none"> • General and trig math • Timers and counters • Bitwise logical • Advanced process • Advanced drive 	<ul style="list-style-type: none"> • Step/action with embedded structured text • Transition with structure text comparisons • Simultaneous and selection branches • Stop element 	<ul style="list-style-type: none"> • General and trig math • Timers and counters • Bitwise logical • Array management • Diagnostic • ASCII manipulation • Specialty CPU control • Motion control • Advanced process • Advanced drive
Editor style	<ul style="list-style-type: none"> • Graphical rungs • Unlimited rungs 	<ul style="list-style-type: none"> • Graphical, free-form drawing • Unlimited sheets 	<ul style="list-style-type: none"> • Graphical, free-form drawing • Unlimited grid space 	<ul style="list-style-type: none"> • Textual • Unlimited lines
Monitoring	<ul style="list-style-type: none"> • Rung animation • Data value animation • Force status 	<ul style="list-style-type: none"> • Output and input pin data value animation 	<ul style="list-style-type: none"> • Active steps animation • Auto display scroll • Branch/transition force status 	<ul style="list-style-type: none"> • Tag watch pane • Context coloring
Comments	<ul style="list-style-type: none"> • Tag • Rung 	<ul style="list-style-type: none"> • Tag • Text box 	<ul style="list-style-type: none"> • Tag • Text box • Embedded structured text comments that are stored in CPU 	<ul style="list-style-type: none"> • Multi-line • End if line • Comments that are stored in CPU

Programming Methods

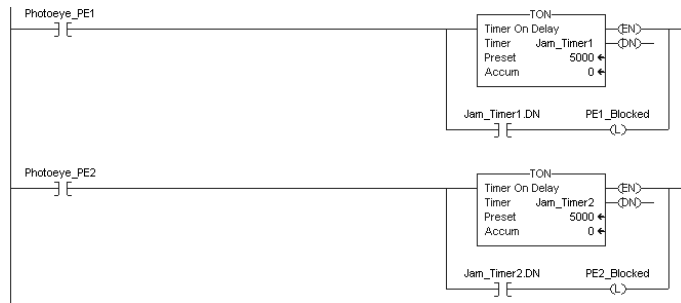
The capabilities of the Logix5000 controllers make different programming methods possible. There are trade-offs to consider when selecting a programming method.

Inline Duplication

Benefits

- Uses more memory
- Fastest execution time because all tag references are defined before runtime
- Easiest to maintain because rung animation matches tag values
- Requires more time to create and modify

Write multiple copies of the code with different tag references.

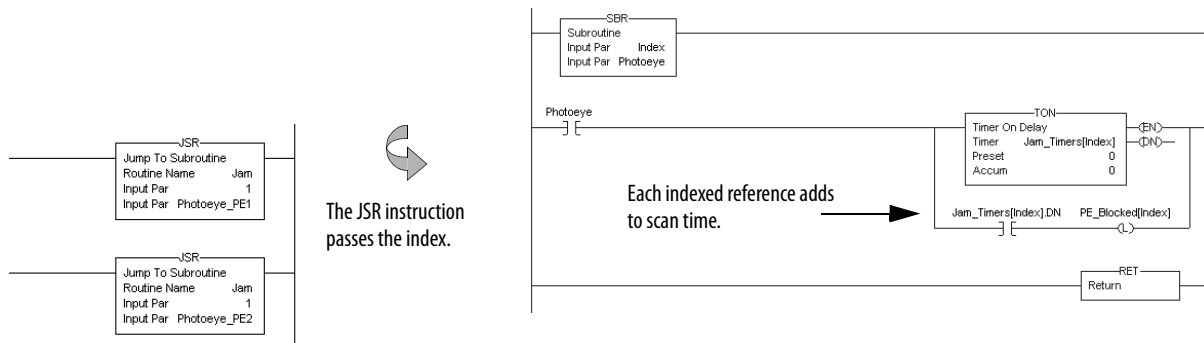


Indexed Routine

Benefits

- One copy of code is faster to develop
- Slowest execution time because all tag references are calculated at run time
- Can be difficult to maintain because the data monitor is not synchronized to execution

Write one copy of code and use indexed references to data stored in arrays.

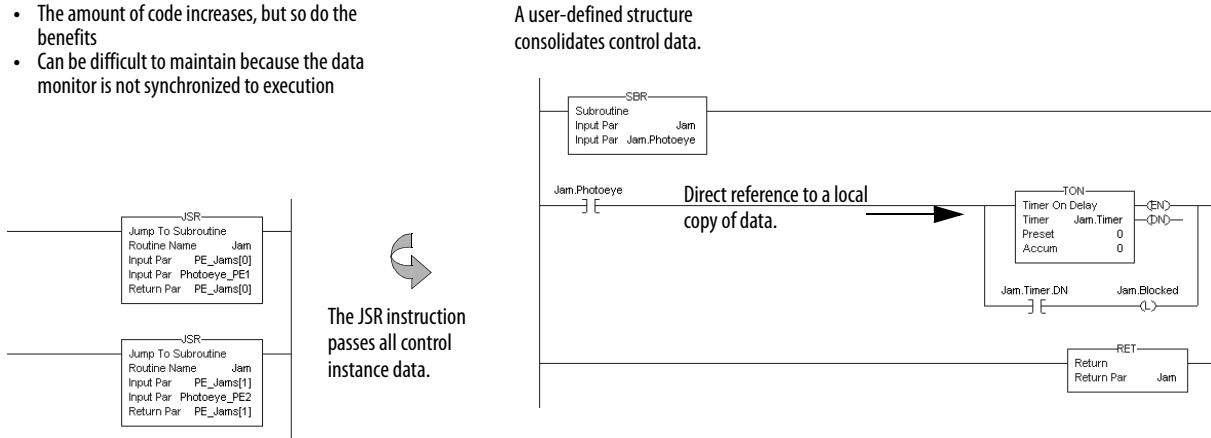


Buffered Routine

Benefits

- One copy operation can occur faster than multiple index offsets
- Eliminates the need to calculate array offsets at run time
- The amount of code increases, but so do the benefits
- Can be difficult to maintain because the data monitor is not synchronized to execution

Copy the values of an array into tags to directly reference these buffer tags.



Controller Prescan of Logic

On transition to Run mode, the controller prescans logic to initialize instructions. The controller resets all state-based instructions, such as outputs (OTE) and timers (TON). Some instructions also perform operations during prescan. For example, the ONSR instructions turns off the storage bit. For information on prescan, see the following resources:

- Logix5000 Controllers General Instructions Reference Manual, publication [1756-RM003](#).
- Logix5000 Controllers Process Control and Drives Instructions Reference Manual, publication [1756-RM006](#).

During prescan, input values are not current and outputs are not written.

Prescan Affects	Description
Relay ladder logic	The controller resets non-retentive I/O and internal values.
Function block diagram logic	Along with resetting non-retentive I/O and internal values, the controller also clears the EnableIn parameter for every function block diagram.
Structured text logic	The controller resets bit tags and forces numeric tags to zero (0). Use the bracketed assignment operator ([:=]) to force a value to be reset during prescan. If you want a tag that is left in its last state, use the non-bracketed assignment operator (:=).
Sequential function chart logic	Embedded structured text follows the same rules as listed previously.

Prescan differs from first scan in that the controller does not execute logic during prescan. The controller executes logic during first scan. The controller sets S:FS for one scan:

- During the first scan that follows prescan.
- During the first scan of a program when it has been uninhibited.
- Each time a step is first scanned (when step.FS is set). You can view the S:FS bit being set only from the logic that is contained in actions that execute during the first scan of their parent step (N, L, P, and P1).

Add-On Instruction Prescan Logic

An Add-On Instruction prescan logic executes after the main logic executes in Prescan mode. Use the prescan logic to initialize tag values before execution. For example, set a PID instruction to Manual mode with a 0% output before its first execution.

When an Add-On Instruction executes in Prescan mode, any required parameters have their data passed.

- Values are passed to Input parameters from their arguments in the instruction call.
- Values are passed out of Output parameters to their arguments defined in the instruction call.

Controller Postscan of SFC Logic

SFCs support an automatic reset option that performs a postscan of the actions that are associated with a step once a transition indicates that the step is completed. Also, every Jump to Subroutine (JSR) instruction causes the controller to postscan the called routine. During this postscan:

- Output energize (OTE) instructions are turned off and non-retentive timers are reset.
- In structured text code, use the bracketed assignment operator (`[:=]`) to have tags reset.
- In structured text code, use the non-bracketed assignment operator (`:=`) to have tags that are left in their last state.
- Selected array faults, that is, 4/20 and 4/83, can be suppressed. When the fault is suppressed, the controller uses an internal fault handler to clear it. Clearing the fault causes the postscan process to skip the instruction containing the fault and continue with the next instruction. This occurs only when SFC instructions are configured for automatic reset.

Add-On Instruction Postscan Logic

When an Add-On Instruction is called by logic in an SFC Action and the Automatic Reset option is set, the Add-On Instruction executes in Postscan mode. An Add-On Instruction postscan routine executes after the main logic executes in Postscan mode. Use the postscan logic to reset internal states and status values or to disable instruction outputs when the SFC action completes.

Timer Execution

Timers in the PLC, SLC, and Logix5000 controllers all store off a portion of the real-time clock each time they are scanned. The next time through, they compare this stored value against the current clock and then adjust the ACC value by the difference.

PLC/SLC Controller	Logix5000 Controller
<p>In a PLC/SLC controller, the timers stores 8 bits at 10 ms/bit. This lets 2.56 seconds ($2^8 / 100$) of padding before a timer overlaps.</p> <p>If program execution skips timers, it appears as if the timers pause. Actually, the timers are overrunning themselves. Depending on when the timer logic next executes, the lost time varies ranges from 0...2.56 seconds.</p>	<p>A Logix5000 controller uses native 32-bit data, so there is more space to store the time. The timer stores 22 bits at 1 ms/bit, which equates to 69.905 minutes ($2^{22} / 1000$ ms per second / 60 seconds per minute).</p> <p>If program execution skips timers, it takes longer than in PLC/SLC controllers to overrun the timers. This results in a larger jump in lapsed time when the timer code next executes.</p>

Program execution can skip executing timers due to the following:

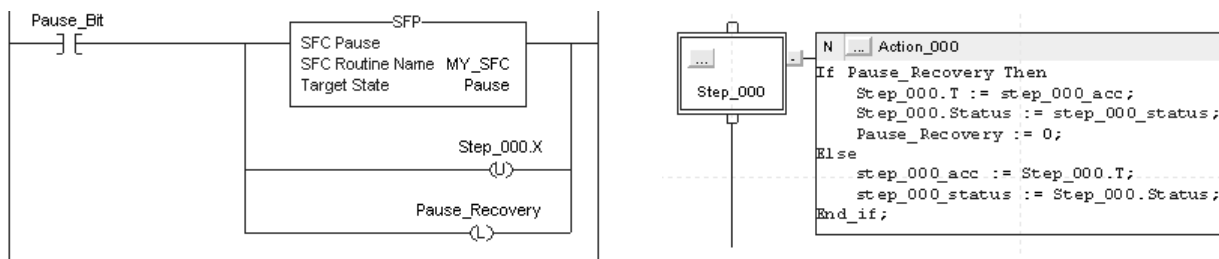
- Subroutine not being called
- Jumping over code
- SFC action
- Inactive SFC step
- Event or periodic task not executing
- Equipment phase state routines

SFC Step Timer Execution

An SFC step timer stores the clock time each time the step executes. On subsequent scans of the step, the controller compares the current clock time with the last scan and updates the step timer's ACC by the difference.

When you pause an SFC and then release the SFC, the step timer jumps forward by the duration of the pause. If you want a step timer to remain at its position during a pause:

- Latch a recovery bit when the chart pause is released.
- Add an action to the step to store the step timer's .ACC value and restore that value when the pause recovery bit is set.



Edit an SFC Online

Firmware revision 13 adds support for editing SFCs online. When you edit an SFC online, the software initially makes the changes in the offline project. When you accept the changes, they are downloaded to the controller. If you transition the controller to test or untest edits, the controller resets the SFC and starts execution at the initial step. If you edit an SFC online, do the following:

- Plan when you test or untest edits to coincide with the SFC executing the initial step.
- Place structured text logic in subroutines to minimize the impact of online edits.
- Use an SFR instruction to shift SFC execution to the desired step programmatically.

In some cases, this can result in the SFC being out of sync with the equipment. Program logic in the initial step to check the last state and use an SFR instruction to change to the appropriate step, if needed. One method is to set an index number in an action of each step. Then when the restart occurs, use the SFR instruction to jump to appropriate step based on the index value.

As of firmware revision 18, these online edits to an SFC no longer reset the SFC to the initial step:

- Modified structured text in actions and transitions
- Physically moved steps, actions, and transitions on SFC sheets without changing the wiring
- Added, deleted, or modified text and description boxes
- Modified indicator tags

Notes:

Modular Programming Techniques

Modular programming guidelines support the delivery of standardized programming structures, conventions, configurations, and strategies. The goal of modular programming is to provide consistency.

- Faster and easier development of application software
- Faster and easier testing of application software
- More reliable application software
- Improved maintenance and operation of application software
- Improved interoperability with other equipment and systems

Guidelines for Code Reuse

Guideline	Description
Use user-defined data types (UDTs) to group data.	<p>Within a UDT:</p> <ul style="list-style-type: none"> • You can mix data types. • The tag names that you assign self-document the structure.
Use Add-On Instructions to create standardized modules of code for reuse across a project.	<p>Use an Add-On Instruction to:</p> <ul style="list-style-type: none"> • Encapsulate specific or focused operations, such as a Motor or Valve action. A Conveyor or Tank action is better managed as a routine. • Create extensions to the base controller instructions. For example, create an Add-On Instruction to execute an SLC 500 or PLC controller instruction not available in the Logix5000 controllers. • Encapsulate an instruction from one language for use in another language. For example, create a function block PIDE instruction for use in relay ladder.
Use program parameters to share data between programs.	<p>Program parameters:</p> <ul style="list-style-type: none"> • Are publicly accessible outside of the program. • Support external HMI external access on an individual basis for each parameter. <p>Direct access lets the user reference program parameters in logic without configuring parameters in the local program. For example, if Program A has an output parameter that is called Tank_Level, Program B can reference the Tank_Level parameter in logic without creating a corresponding parameter to connect to Program A.</p>
Use partial import/export programs, routines, Add-On Instructions, and code segments to create libraries of reusable code.	<p>Partial import and export of routines and programs:</p> <ul style="list-style-type: none"> • Provides more control over the scope of what is extracted from the project. • Provides reusable code for larger machine, cell, or unit control. • Promotes collaboration between multiple engineers, code standardization, and reuse. <p>The export .LSX file includes all pertinent information, including program configuration, code, user-defined data-types, tags, and descriptions, in an XML-formatted, ASCII text file. Use partial import/export to:</p> <ul style="list-style-type: none"> • Distribute code separately from the project .ACD file. • Edit and create programs and routines by using other editing tools.
Use subroutines to reuse code within a program.	<p>Subroutines:</p> <ul style="list-style-type: none"> • Can be created and used in standard and safety applications. • Pass User-Defined Structures (UDT). • Pass all input and output Parameters by value. • Subroutines require the most overhead to pass parameters when called. • Can only be called from within the program they reside.

Naming Conventions

The following conventions are guidelines to help make an engineering library more reusable by other developers. These guidelines also help the resulting applications have a more consistent look and feel.

- Names that are meaningful (and readable) to people who use the application as a later date are most effective.
- Names use controller memory and have limited length, so keep them short by using abbreviations and acronyms. Use mixed case rather than underscore characters to indicate words.
- When you use acronyms, use those that are common or provided by industry standards.

Names for controller logic components must follow these guidelines.

- The name must start with a letter, either upper or lower case
- The name can contain as many as 40 characters; any mix of upper case letter, lower case letters, numbers, and underscore characters
- Case is not significant. The controller interprets Mix_Tank the same and mix_tank. However, the software displays the case as entered
- Underscores are significant. The controller interprets AB_CD as unique from A_BCD
- You cannot have two or more underscore characters in a row
- The name cannot end with an underscore.

Component Name	Recommendations														
Controller	Area, unit, or units the controller controls, underscore, type of controller Example: <table border="1" data-bbox="646 1125 1271 1192"> <tr> <td data-bbox="646 1125 954 1192">Area/Unit + Type</td> <td data-bbox="959 1125 1271 1192">Controller Name: Mixing:ControlLogix</td> </tr> </table>	Area/Unit + Type	Controller Name: Mixing:ControlLogix												
Area/Unit + Type	Controller Name: Mixing:ControlLogix														
Controller project	Controller name, the letter C, 1-digit major revision number, underscore, 2-digit minor revision number Example: <table border="1" data-bbox="646 1272 1385 1339"> <tr> <td data-bbox="646 1272 1016 1339">Project in controller Mixing_CLX, Major Revision 1, Minor Revision 02</td> <td data-bbox="1021 1272 1385 1339">Application Name: Mixing_CLX_C2_092.ACD</td> </tr> </table> <p data-bbox="589 1356 1471 1402">Increment the minor revision number for any documented engineering change according to the code in the controller (for example, the code for minor process or equipment changes).</p> <p data-bbox="589 1409 1471 1455">Increment the major revision number for any documented engineering change according to the code in the controller that implements a design change (for example, code that enhances or reduces controller functionality).</p>	Project in controller Mixing_CLX, Major Revision 1, Minor Revision 02	Application Name: Mixing_CLX_C2_092.ACD												
Project in controller Mixing_CLX, Major Revision 1, Minor Revision 02	Application Name: Mixing_CLX_C2_092.ACD														
Tag	Prefix with the abbreviation of the type of tag Examples: <table border="1" data-bbox="646 1539 1341 1797"> <tr> <td data-bbox="646 1539 992 1577">Interprocessor communication tag</td> <td data-bbox="997 1539 1341 1577">IPC_</td> </tr> <tr> <td data-bbox="646 1583 992 1621">Input tag</td> <td data-bbox="997 1583 1341 1621">I_</td> </tr> <tr> <td data-bbox="646 1627 992 1665">Output tag</td> <td data-bbox="997 1627 1341 1665">O_</td> </tr> <tr> <td data-bbox="646 1671 992 1709">Remote I/O tag</td> <td data-bbox="997 1671 1341 1709">RIO_</td> </tr> <tr> <td data-bbox="646 1715 992 1753">Control module class tag</td> <td data-bbox="997 1715 1341 1753">Device ID_</td> </tr> <tr> <td data-bbox="646 1759 992 1797">Equipment module class tag</td> <td data-bbox="997 1759 1341 1797">EM_</td> </tr> <tr> <td data-bbox="646 1803 992 1841">Equipment phase class tag</td> <td data-bbox="997 1803 1341 1841">EP_</td> </tr> </table>	Interprocessor communication tag	IPC_	Input tag	I_	Output tag	O_	Remote I/O tag	RIO_	Control module class tag	Device ID_	Equipment module class tag	EM_	Equipment phase class tag	EP_
Interprocessor communication tag	IPC_														
Input tag	I_														
Output tag	O_														
Remote I/O tag	RIO_														
Control module class tag	Device ID_														
Equipment module class tag	EM_														
Equipment phase class tag	EP_														

Component Name	Recommendations																																								
I/O or communication module	<p>Controller name, underscore, abbreviation of rack location (L=local, R=remote), underscore, the letter S, 2-digit slot number, underscore, abbreviation of function</p> <p>Example Functions:</p> <table border="1" data-bbox="651 327 1398 810"> <tbody> <tr><td>Analog input</td><td>AI</td></tr> <tr><td>Analog output</td><td>AO</td></tr> <tr><td>Discrete input</td><td>DI</td></tr> <tr><td>Discrete output</td><td>DO</td></tr> <tr><td>Analog input/output combination</td><td>AIO</td></tr> <tr><td>Discrete input/output combination</td><td>DIO</td></tr> <tr><td>Analog/discrete input/output combination</td><td>ADIO</td></tr> <tr><td>Serial data</td><td>SIO</td></tr> <tr><td>Motion data</td><td>MIO</td></tr> <tr><td>DeviceNet data</td><td>DNET</td></tr> <tr><td>EtherNet/IP data</td><td>ENET</td></tr> <tr><td>ControlNet</td><td>CNET</td></tr> <tr><td>Remote I/O data</td><td>RIO</td></tr> </tbody> </table> <p>Examples:</p> <table border="1" data-bbox="651 890 1398 1297"> <tbody> <tr><td>Mixer123 Controller, Local chassis, Slot 4, Analog Output</td><td>Module Name: M123_CLX_L00_S04_AO</td></tr> <tr><td>Mixer123 Controller, Local chassis, Slot 12, Discrete Output</td><td>Module Name: M123_CLX_L00_S12_DO</td></tr> <tr><td>Mixer123 Controller, Remote chassis #1, Slot 1, Analog Input</td><td>Module Name: M123_CLX_R01_S01_AI</td></tr> <tr><td>Mixer123 Controller, Remote chassis #1, Slot 2, Analog Output</td><td>Module Name: M123_CLX_R01_S02_AO</td></tr> <tr><td>Mixer123 Controller, Remote chassis #2, Slot 5, Discrete Input</td><td>Module Name: M123_CLX_R02_S05_DI</td></tr> <tr><td>Mixer123 Controller, Remote chassis #2, Slot 6, Discrete Output</td><td>Module Name: M123_CLX_R02_S06_DO</td></tr> <tr><td>Mixer123 Controller, Local chassis, Slot 5, Remote I/O</td><td>Module Name: M123_CLX_R02_S06_RIO</td></tr> </tbody> </table>	Analog input	AI	Analog output	AO	Discrete input	DI	Discrete output	DO	Analog input/output combination	AIO	Discrete input/output combination	DIO	Analog/discrete input/output combination	ADIO	Serial data	SIO	Motion data	MIO	DeviceNet data	DNET	EtherNet/IP data	ENET	ControlNet	CNET	Remote I/O data	RIO	Mixer123 Controller, Local chassis, Slot 4, Analog Output	Module Name: M123_CLX_L00_S04_AO	Mixer123 Controller, Local chassis, Slot 12, Discrete Output	Module Name: M123_CLX_L00_S12_DO	Mixer123 Controller, Remote chassis #1, Slot 1, Analog Input	Module Name: M123_CLX_R01_S01_AI	Mixer123 Controller, Remote chassis #1, Slot 2, Analog Output	Module Name: M123_CLX_R01_S02_AO	Mixer123 Controller, Remote chassis #2, Slot 5, Discrete Input	Module Name: M123_CLX_R02_S05_DI	Mixer123 Controller, Remote chassis #2, Slot 6, Discrete Output	Module Name: M123_CLX_R02_S06_DO	Mixer123 Controller, Local chassis, Slot 5, Remote I/O	Module Name: M123_CLX_R02_S06_RIO
Analog input	AI																																								
Analog output	AO																																								
Discrete input	DI																																								
Discrete output	DO																																								
Analog input/output combination	AIO																																								
Discrete input/output combination	DIO																																								
Analog/discrete input/output combination	ADIO																																								
Serial data	SIO																																								
Motion data	MIO																																								
DeviceNet data	DNET																																								
EtherNet/IP data	ENET																																								
ControlNet	CNET																																								
Remote I/O data	RIO																																								
Mixer123 Controller, Local chassis, Slot 4, Analog Output	Module Name: M123_CLX_L00_S04_AO																																								
Mixer123 Controller, Local chassis, Slot 12, Discrete Output	Module Name: M123_CLX_L00_S12_DO																																								
Mixer123 Controller, Remote chassis #1, Slot 1, Analog Input	Module Name: M123_CLX_R01_S01_AI																																								
Mixer123 Controller, Remote chassis #1, Slot 2, Analog Output	Module Name: M123_CLX_R01_S02_AO																																								
Mixer123 Controller, Remote chassis #2, Slot 5, Discrete Input	Module Name: M123_CLX_R02_S05_DI																																								
Mixer123 Controller, Remote chassis #2, Slot 6, Discrete Output	Module Name: M123_CLX_R02_S06_DO																																								
Mixer123 Controller, Local chassis, Slot 5, Remote I/O	Module Name: M123_CLX_R02_S06_RIO																																								

Parameter Name Prefixes

Programming structures, such as Add-On Instructions and programs support parameters for passing values. The convention for prefixes is to abbreviate the function of the parameter to three letters and an underscore, followed by additional text to clarify the specific function.

Parameter Function	Prefix	Description
Command	Cmd_	Designates a command input, either from the operator via the HMI or from the program. Examples: <ul style="list-style-type: none"> • Cmd_Reset: Clear faults and reset the process • Cmd_JogServo: Jog a servo axis • Cmd_FillTank: Fill a tank with a liquid
Configuration	Cfg_	Designates a configuration value for the structure. Enter from the HMI or as part of a recipe. Examples: <ul style="list-style-type: none"> • Cfg_JogDirection: Selects the direction a servo jogs: 0=Positive, 1=Negative • Cfg_BulkFill: Selects the fill rate to use: 0=Slow Rate, 1=Fast Rate • Cfg_UserUnits: Selects the measure of volume to use: 0=mm, 1=m, 2=gal • Cfg_EnableInterlocks: Enable interlock functionality • Cfg_EnablePermissive: Enable permissive functionality
Status	Sts_	Status of the process within the structure. Examples: <ul style="list-style-type: none"> • Sts_Alarm: An alarm condition (such as a HI/LOW alarm) exists within the process • Sts_ER: An error with an instruction execution within the process has been detected • Sts_IndexComplete: The servo index move within the process has completed • Sts_FillInProcess: The tank filling process is underway
Error	Err_	If the Sts_ER bit is on, the Err_ parameter indicates the actual error. This can be either a bit level or value level indication. <ul style="list-style-type: none"> • Bit level error recording supports multiple errors simultaneously, but can require a large number of indicators to support all error states. • Value-based error annunciation supports a large quantity of errors within a single indicator. However, this approach requires that errors are annunciated one at a time. Examples: <ul style="list-style-type: none"> • Err_Value: A non-zero value indicates an error condition • Err_PCamCalcFault: Indicates that an error has occurred in an MCCP
Alarm	Alm_	If the Sts_Alm bit is on, the Alm_ parameter indicates which alarm is occurring. This can be either a bit-level or value-level indication. <ul style="list-style-type: none"> • Bit-level alarming supports multiple alarms simultaneously, but can require a large number of indicators to support all alarm states. • Value-based alarm annunciation supports a large quantity of alarms within a single indicator. However, this approach requires alarms to be annunciated one at a time. Examples: <ul style="list-style-type: none"> • Alm_Value: A non-zero value indicates an alarm condition • Alm_TankHI: Indicates that a HI level condition has been detected within a tank
Input	Inp_	Real-time data used to drive the process. Designates a connection either to a real input point, a control device, or to data received from other processes. Examples: <ul style="list-style-type: none"> • Inp_ServoPosition: Variable providing the input value for a position of a servo • Inp_ServoRegistrationPosition: Input of a the registration position of the servo • Inp_InterlockOK: Input indicating external interlocks are met • Inp_TankLevel: Variable providing the analog input for a tank' level • Inp_TankLevelFillRate
Output	Out_	Real-time data driven from the process. Designates a connection to a real output point, a control device, or to data sent to other processes. Examples: <ul style="list-style-type: none"> • Out_GlueGun1: Output signal to turn of Glue Gun 1 • Out_ServoCorrectionDistance: Output of a servo registration correction distance • Out_OverflowValve: Output signal to open the Overflow Valve • Out_TankLevelError: Output of a difference between target and actual fill level of a tank
Reference	Ref_	Complex data structures that combine input and output data. These structures pass data into a structure, where some process is performed. The results are then loaded back into the structure to be passed out of AOI for use elsewhere. Example: Ref_PositionCamRecovery: Provides the data set for calculating a Position Cam with all offsets factored in, as well as the resulting Position Cam Profile to run in an MAPC instruction

Parameter Function	Prefix	Description
Parameter	Par_	Variables that are received from an external source that can be internal or external to the program. Examples: <ul style="list-style-type: none"> Par_MachineSpeed: Provides a machine's running speed Par_TargetFillLevel: Provides a tank's target fill level
Set point	Set_	Variables received from an operator or HMI and are not part of an external source. Examples: <ul style="list-style-type: none"> Set_MachineMaxSpeed: Provides the setting for a machine's maximum permissible speed Set_TankHILevel: Provides the setting for a tank's HI alarm limit
Value	Val_	Designates a value that might not be the primary output of the structure.
Report	Rpt_	Designates a value that is typically used for reporting.
Information	Inf_	Non-functional data such as an revision level or name for displaying a faceplate.
Ready	Rdy_	Command-ready bits that are typically Booleans calculated inside the control routines to reflect whether the routine let states change commands. Used with HMI faceplates to enable or disable command buttons.
Program Command (optional)	PCmd_	Command input for commands typically issued by the application program. Examples: <ul style="list-style-type: none"> PCmd_ProgReq - Request for Program Mode made by the application (as opposed to Cmd_ProgProgReq) PCmd_AutoReq - Request for Auto Mode made by the application (as opposed to Cmd_ProgAutoReq)
Operator Command (optional)	OCmd_	Command input for commands typically issued by the operator via the HMI. Examples: <ul style="list-style-type: none"> OCmd_ProgReq - Request for Program Mode made by the operator (as opposed to Cmd_OperProgReq) OCmd_AutoReq - Request for Auto Mode made by the operator (as opposed to Cmd_OperAutoReq)

Guidelines to for Subroutines Follow these parameter guidelines for subroutines.

Guideline	Description
Input and Return parameters depend on the subroutine logic.	If the subroutine needs to know the previous state of any Return parameters (the values are used elsewhere in the project), these values should also be Input parameters: <ul style="list-style-type: none"> If the subroutine contains latch/unlatch logic (holding circuits), intended outputs of the subroutine should be passed into and returned from the subroutine. If the subroutine does not contain latch/unlatch logic, intended outputs of the subroutine only need to be returned from the subroutine.
Pass complete timers in and out of subroutines.	If a subroutine needs a timer, pass the complete timer tag to the subroutine as an input and return the complete timer tag as an output. Store the timer in a buffer tag outside of the subroutine.
Create a user-defined tag to pass large numbers Input and Output parameters	Create and pass a UDT if you have several Input and Output parameters to save on execution time. The more parameters you pass, the fewer nested JSRs you can perform.
Data types must match	For each parameter in a SBR or RET instruction, use the same data type (including any array dimensions) as the corresponding parameter in the JSR instruction. Using different data types can produce unexpected results.

Guidelines for User-defined Data Types

A UDT lets you organize or group data logically, so that all of the data associated with a device (such as a pressure transmitter or variable frequency drive) can be grouped together.

- You can mix data types, such as real or floating point values, counters, timers, arrays, Booleans, and other UDTs, within one UDT.
- You can copy a UDT from one project to another, and even from one Logix controller type to another.
- A UDT is self-documenting based on the tag names you assign, and provides a logical representation of parts or sub-systems.

Naming Conventions for User-Defined Data Types

Element	Description
Prefix_	UDT_
UDT name	Function or purpose of the UDT

Examples:

Inventory tracking tag	UDT_InventoryTracking
Clean in place system	UDT_CIP
Two-state valve control module in control module	UDT_CMV2S
Water addition in equipment module	UDT_EM

UDT Member Order

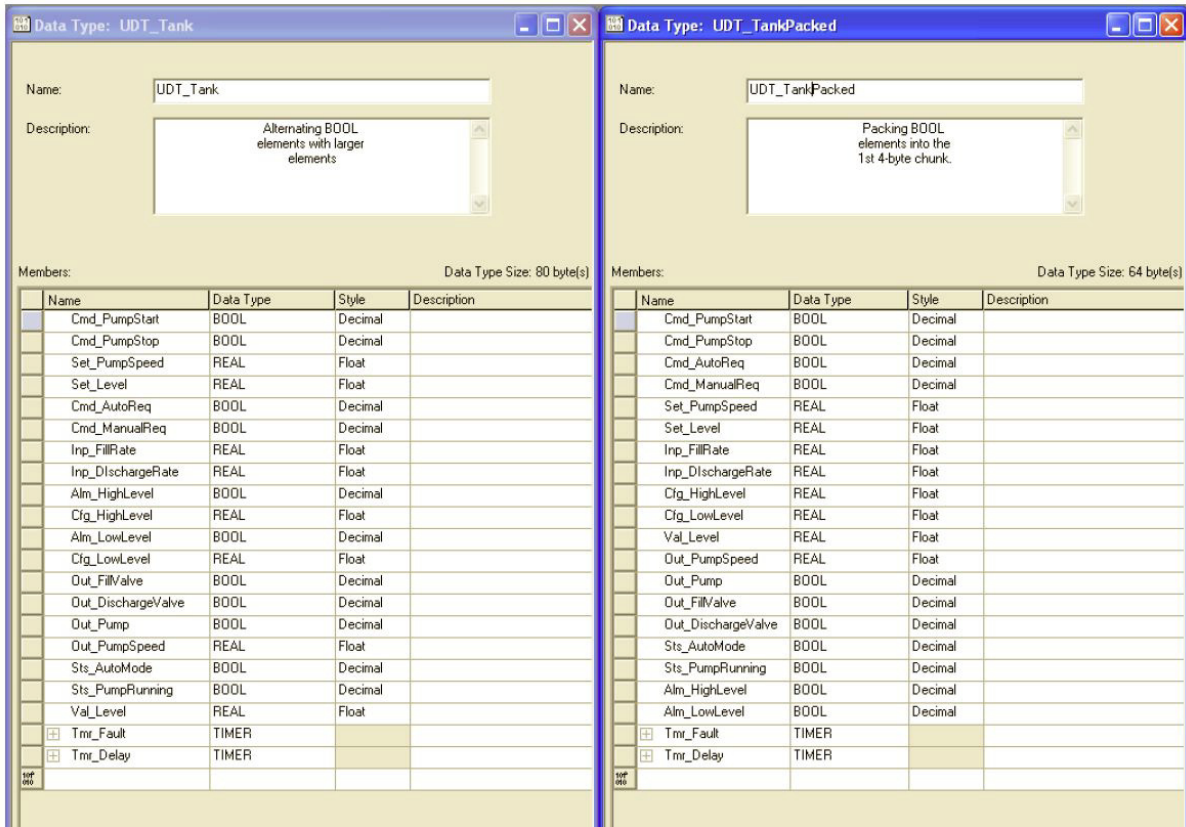
The order in which elements are listed in the UDT can have a significant impact on memory use if several BOOL, INT, or SINT elements are defined. Memory is allocated in 4-byte (32-bit) increments, and every DINT, REAL, STRING, or sub-UDT element always start at the beginning of a 4-byte boundary.

For example, if the first element defined is a BOOL, it uses the first 4 bytes allocated to the UDT. Other BOOLs can be assigned immediately following without consuming any more memory, until the first 4 bytes are consumed. However, if the next element is a DINT, the DINT element allocates another 4 bytes even though the BOOL occupies only a single bit in the first 4 bytes. So for this example, the 31 bits of memory between the BOOL and the start of the DINT are allocated but are not accessible.

- UDT memory is allocated in 4-byte increments.
- Elements that occupy 4 bytes or more always start at a 4-byte boundary. These include DINT, REAL, STRING, any UDT, or any other complex data structure.
- Elements of smaller data types (BOOL, SINT, or INT) start on the next byte boundary that matches its size, so that all the data types in the UDT are fully contained in their respective 4-byte increments. For example, INT elements start on 2-byte boundaries, SINT elements follow at the next byte, and BOOL elements in succession occupy consecutive bits within a byte.

In the following example, the UDT on the left, UDT_Tank, has members arranged by function without regard for memory usage. This makes sense in the context of implementation, because members toward the top are ordinarily used in the software code.

However, the disjointed listing of data types in UDT_Tank consumes 25% more memory than the example UDT on the right, UDT_TankPacked. In UDT_TankPacked, the BOOL members are grouped according to their functionality, with the input BOOLs grouped at the top and the output BOOLs grouped at the bottom. As a result, the data type size is reduced from 80 bytes to 64 bytes.



Guidelines for Add-On Instructions

An Add-On Instruction encapsulates commonly used functions or device controls. It is not intended for use as a high-level hierarchical design tool. Once an Add-On Instruction is defined in a project, it behaves similarly to the built-in instructions that are already available in the programming software. The AOI appears on the instruction toolbar and in the instruction browser.

Guideline	Description
<p>Create Add-On Instructions in relay ladder, function block diagram, or structured text languages.</p>	<p>Supports all Add-On Instructions and most built-in instructions. Excludes JSR/SBR/RET, JXR, FOR/BRK (relay ladder), SFR, SFP, SAR, IOT, and EVENT instructions.</p> <p>GSV/SSV instructions in an Add-On Instruction cannot reference the Module, Message, Axis, Motion Group, or Coordinate System class names.</p> <p>Add-On Instructions support function block, relay ladder, and structured text programming languages. Each of the Add-On Instruction logic areas can be any language. For example, the main logic can be function block and the prescan logic can be relay ladder.</p> <p>You can nest Add-On Instructions seven levels deep.</p> <p>As of RSLogix 5000 software, version 18, you can create safety Add-On Instructions in a safety task.</p>
<p>An Add-On Instruction supports parameters:</p> <ul style="list-style-type: none"> • Input (copied in) • Output (copied out) • InOut (passed by reference) 	<ul style="list-style-type: none"> • Limited to 512 total: Input parameter + Output parameter + local tags (no limit on the number of InOut parameters) • 2 MB maximum data instance (parameters and locals) • Alarm, axis, axis group, coordinate system, message, motion group, and produced/consumed tags must exist at the program or controller scope and passed as an InOut parameter • Can include references to controller-scoped tags, program-scoped tags, and immediate values. • Input and Output parameters are limited to atomic (BOOL, SINT, INT, DINT, REAL) data types. Use the InOut parameter for LINT, user-defined, and structure data types. • DINT data types provide optimal execution. • Default values of parameters and local tags are used to initialize the data structure when a tag is created of the instruction's data type. When an existing parameter or local tag's default value is modified, the existing tag instances for that instruction are not updated. When a parameter or local tag is added to the instruction definition, the tag's default value is used in the existing tags.
<p>Create and modify offline only.</p>	<p>Online operation supports monitoring.</p> <p>Modifications to Add-On Instructions are made offline. Make changes once to the Add-On Instruction definition to affect all instances.</p>
<p>An Add-On Instruction executes like a routine.</p>	<p>A task with a higher execution priority can interrupt an Add-On Instruction. Use a UID/UIE instruction pair to make sure an Add-On Instruction's execution is not interrupted by a higher priority task.</p> <p>If you have many parameters or specialized options, consider multiple Add-On Instructions</p> <p>Calling many Add-On Instructions impacts scan time</p>
<p>The code within an Add-On Instruction can access data that is specified only via parameters or defined as local.</p>	<p>Copy the local data to a parameter if you want to programmatically access it outside of an Add-On Instruction.</p>
<p>Use optional Scan mode logic to set up, initialize, or reset the Add-On Instruction code.</p>	<p>An Add-On Instruction can have logic along with the main logic for the instruction.</p> <ul style="list-style-type: none"> • Prescan logic executes on controller startup. • Postscan logic executes on SFC Automatic reset. • EnableInFalse logic executes when rung condition is false.
<p>Apply code signatures to Add-On Instructions for revision control.</p>	<p>Add-On Instructions can be sealed with a code signature, as of RSLogix 5000 software, version 18. Use the code signature for revision control and to identify any changes. For safety controllers, the signature can be used to get TUV certification for a safety Add-On Instruction. For more information, see the Logix5000 Controllers Add-On Instructions Programming Manual, publication 1756-PM010.</p>

Add-On Instruction Design Concepts

To be sure that specific data is passed into or out of the add-on instruction, use a required parameter. A required parameter must be passed as an argument in order for a call to the instruction for verification. To pass a required parameter in ladder diagrams and in structured text, specify an argument tag for the parameter.

- In a function block diagram, required Input parameters and Output parameters must be wired.
- In a ladder diagram, InOut parameters must have an argument tag.
- If a required parameter lacks an associated argument, the routine that contains the call to the add-on instruction does not verify.

Naming Conventions for Add-On Instructions

Component Name	Recommendations				
Add-On Instruction	<p>Start with the application name. Add a variant name, if applicable. Capitalize the first letter in all words in the name.</p> <p>Example:</p> <table border="1" style="margin-left: 40px;"> <tr> <td>PCam profile display</td> <td>PCamProfileDisplay</td> </tr> </table> <p>Suffix with underscore AOI, if space permits.</p> <p>Example:</p> <table border="1" style="margin-left: 40px;"> <tr> <td>PCam profile display</td> <td>PCamProfileDisplay_AOI</td> </tr> </table>	PCam profile display	PCamProfileDisplay	PCam profile display	PCamProfileDisplay_AOI
PCam profile display	PCamProfileDisplay				
PCam profile display	PCamProfileDisplay_AOI				

Comparison of Subroutines and Add-On Instructions

Comparison	Subroutine	Add-On Instructions
Accessibility	Within program (multiple copies)	Anywhere in controller (single copy)
Parameters	Pass by value	Pass by value or reference via InOut
Numeric parameters	No conversion, you must manage	Automatic data type conversion for Input and Output parameters InOut parameters must match declared type exactly
Parameters data types	Atomic, arrays, structures	<ul style="list-style-type: none"> • Atomic data types as In or Out parameters • LINT, user-defined, and structure data types as InOut parameters
Parameter checking	None, you must manage	Verification checks
Data encapsulation	All data at program or controller scope (accessible to anything)	Local data is isolated (only accessible within instruction)
Monitor/debug	Logic that is animated with mixed data from multiple calls	Logic that is animated with data from one calling instance
Supported programming languages	FBD, LD, SFC, ST	FBD, LD, ST
Callable from	FBD, LD, SFC, ST	FBD, LD, SFC, ST
Protection	Locked and View Only	Locked and View Only
Documentation	Routine, rung, textbox, line	Instruction description, revision information, vendor, rung, textbox, line, extended help
Execution performance	<ul style="list-style-type: none"> • JSR/SBR/RTN add overhead • All data is copied 	<ul style="list-style-type: none"> • Call is more efficient • InOut passed by reference

Comparison	Subroutine	Add-On Instructions
Memory use	Compact	<ul style="list-style-type: none"> • Call requires more memory • All references need additional memory
Edit	Both code and data can be modified offline and online in a running controller	Code modifications are limited to offline in the project file and require a new download Data values associated can be modified online and offline
Import/export	All routines are imported/exported in the full project .LSK file (protected routines can be excluded or encrypted) Individual LD rungs and references and tags/UDTs can be imported/exported via the .LSX file	All Add-On Instructions are imported/exported in the full project .LSK file (protected instructions can be excluded or encrypted) Individual Add-On Instruction definitions and code are imported/exported via the .LSX file

Comparison of Partial Import/Export and Add-On Instructions

Comparison	Partial Import/Export	Add-On Instructions
Logic	Any program, equipment phase, routine, Add-On Instruction, or user-defined data type in the project can be imported/exported via .LSX file.	Create once (single copy) and use anywhere in the same controller project.
Controller accessibility	Import on-line with a running controller: <ul style="list-style-type: none"> • Add programs, routines, and Add-On Instructions • Existing programs and routines can be replaced • Create tags and UDTs • Name collisions are detected automatically and you are prompted to rename or bind to existing components • The data values in the controller are maintained and new tags have their values initialized from the import file 	Existing Add-On Instructions can only be edited offline. New Add-On Instructions can be created online or offline.
Logic checking	You resolve conflicts on import.	The software verifies the components that you add to Add-On Instruction as you create it.
Data	Editing member definitions of an Add-On Instruction maintains the values that are assigned to the parameters when: <ul style="list-style-type: none"> • Inserting, adding, or deleting members • Rearranging (moving) members • Renaming members • Changing the data types of members Values for members that are both renamed and moved in the same operation are not to be maintained.	Local data is isolated (only accessible within the instruction).

Guidelines for Program Parameters

Program parameters define a data interface for programs to facilitate data sharing. Data sharing between programs can be achieved either through pre-defined connections between parameters or directly through a special notation. Unlike local tags, all program parameters are publicly accessible outside of the program. Additionally, HMI external access can be specified on individual basis for each parameter.

Standard (non-Safety) parameters can be created, edited, and deleted while online with the controller. The following exceptions apply:

- Parameters cannot be deleted while online if they are connected/bound to other parameters, or if the control logic references them.
- InOut parameters cannot be deleted while online
- InOut bindings can only be changed online through a Partial Import Online (PIO) operation

A safety parameter cannot be connected with or bound to a standard parameter or controller scoped tag. A safety connection cannot be created, modified, or deleted in a Safety Locked project. Input, Output, and Public parameters support the External Access attribute. InOut parameters do not.

Program Parameter	Description
Input	<ul style="list-style-type: none"> • Input parameters (including members) can only support ONE connection. Only one source can be delivering the value to the input parameter. • Input Parameter values are refreshed before each scan of a program. The values do not change during the logic execution so you do not need to write code to buffer inputs. • A program can write to its own input parameters. • Data values for Output parameters that are connected to controller scope tags or Public parameters are copied after the scan of a program. In a project with multiple tasks, the data copy for a parameter that is of type BOOL, SINT, INT, DINT, LINT, or REAL will not be interrupted. A task switch can interrupt the data copy from an Output parameter to a controller scope tag or Public parameter, or any other predefined or user-defined data type.
Output	<ul style="list-style-type: none"> • Output parameters (including members) can support multiple connections. For example, lets assume you have a BOOL input parameter in Program A and Program B named Input1a and Input1b. You can connect an output parameter in Program C to Input1a AND Input1b. As stated earlier, this is often referred to as fanning. • Output Parameter values are refreshed AFTER each scan of a program. Updated output parameter values are NOT available to the parameters connected to that output parameter until the program execution is complete. • Output parameters that are connected to Public parameters or controller scope tags are copied (pushed) at the end of the program execution. • An Output parameter can ONLY be connected to an InOut parameter if both the Output and InOut parameters are configured as Constants.
InOut	<ul style="list-style-type: none"> • InOut parameters can only support ONE connection. You cannot configure connections to any member of an InOut parameter. • InOut parameters are passed by REFERENCE, which means they simply point to the base tag. In other words, when an InOut parameter is used in logic, the current value of the parameter that is connected to the InOut Parameter is used. • An InOut parameter can ONLY be connected to an Output parameter if both the Output and InOut parameters are configured as Constants. See the tool tip for Output Parameters for a more detailed explanation. • InOut parameters CANNOT be changed online, unless using the Partial Import Online (PIO).
Public	<ul style="list-style-type: none"> • Public parameters can support MULTIPLE connections. You can configure connections to the base Public parameter or any member of a Public parameter. This includes User-Defined Structures. • Public parameters are updated when the source is updated. In other words, when a Public parameter value updates, it is immediately available to any higher priority tasks that are connected to that parameter. • Public parameters can be aliased to Controller Scope Tags. If this functionality is desired, remember that the alias update is asynchronous to program execution. The public parameter contains the real-time value of the controller scope tag.

Comparison of Program Parameters and Add-On Instructions

Comparison	Program Parameters	Add-On Instructions
Accessibility	Within program (multiple copies)	Anywhere in controller (single copy)
Parameters	Input / Output (pass by value), InOut (pass by reference), Public (pass by value)	Input / Output (pass by value), InOut (pass by reference)
Numeric parameters	<ul style="list-style-type: none"> Automatic data type conversion for Input and Output parameters InOut parameters must match declared type exactly 	<ul style="list-style-type: none"> Automatic data type conversion for Input and Output parameters InOut parameters must match declared type exactly
Parameters data types	Atomic, strings, arrays, structures	<ul style="list-style-type: none"> Atomic data types as In or Out parameters LINT, user-defined, and structure data types as InOut parameters
Parameter checking	None, user must manage	Verification checks
Data encapsulation	All data at program or controller scope (accessible to anything). Programs can talk directly and exchange data between them. Local tags remain private to the Program. Cannot access Local Tags, only the parameters.	Local data is isolated (only accessible within instruction)
Monitor/debug	Online editable.	Logic that is animated with data from one calling instance
Supported programming languages	FBD, LD, SFC, ST	FBD, LD, ST
Callable from	FBD, LD, SFC, ST	FBD, LD, SFC, ST
Protection	—	Locked and View Only
Documentation	—	Instruction description, revision information, vendor, rung, textbox, line, extended help
Execution performance	<ul style="list-style-type: none"> Programs can talk directly and exchange data between them. InOut passed by reference 	<ul style="list-style-type: none"> Call is more efficient InOut passed by reference
Memory use	Compact. One Public parameters can be connected or bound to multiple Input, Output or InOut parameters to form a shared memory space.	<ul style="list-style-type: none"> Call requires more memory All references need additional memory
Edit	Online editable, and supports sub-element connections. Copy / Paste Programs without disturbing parameter configuration.	Code modifications are limited to offline in the project file and require a new download Data values associated can be modified online and offline

Address Data

Logix5000 controllers support IEC 61131-3 atomic data types, such as BOOL, SINT, INT, DINT, LINT, and REAL. The controllers also support compound data types, such as arrays, predefined structures (such as counters and timers), and user-defined structures.

Data Type	Description	
Atomic data type (BOOL, SINT, INT, DINT, REAL)	Benefit <ul style="list-style-type: none"> • Individual names • No limit to the number of tags • Tag Editor and Data Monitor can filter individual tags and display any references • Always listed alphabetically in the Tag Editor and Data Monitor • Full alias tag support (both the base tag and its bits) • Can be added when programming online • Supported as In or Out parameter in an Add-On Instruction 	Consideration <ul style="list-style-type: none"> • Require more communication overhead and, potentially, more controller memory than compound data types • Can only change the data type of a flag when programming offline • The root tag is listed alphabetically in the Tag Editor and Data Monitor, but the structure members are listed in the order in which they were defined in the structure
Special-use atomic data type (LINT)	Benefit <ul style="list-style-type: none"> • 64-bit integer value to store date and time values • Data monitor display radix for Date and Time lets you display a LINT value as year, month, day, hours, minutes, seconds, microseconds 	Consideration <ul style="list-style-type: none"> • Limited instruction support: GSV, SSV, ALMD, ALMA, COP, and CPS • For math operations or comparisons, copy the LINT value into a pair of DINTs and then manipulate through code • Limited to InOut parameter in an Add-On Instruction
Compound data type (array, structure)	Benefit <ul style="list-style-type: none"> • Specific names and user-defined organization are available • Consolidates information in controller memory • Optimizes communication time and memory impact • Arrays can be dynamically indexed • Can create arrays when programming online • Alias support for user-defined structures, members of an array, and bits of a member 	Consideration <ul style="list-style-type: none"> • 2 MB data limit per user-defined structure or array • User-defined structures are padded to enforce 32-bit data alignment • Alias tags cannot point to the root tag of an array • Tag Editor and Data Monitor filtering limited • Can only create or change a user-defined structure when programming offline • Can only change an array when programming offline • Limited to InOut parameter in an Add-On Instruction

The Logix CPU reads and manipulates 32-bit data values. The minimum memory allocation for data in a tag is 4 bytes. When you create a standalone tag that stores data that is less than 4 bytes, the controller allocates 4 bytes, but the data only fills the part that it needs.

Data Type	Bits								
	64...32	31	16	15	8	7	1	0	
BOOL	Not allocated	Allocated but not used						0 or 1	
SINT	Not allocated	Allocated but not used				-128...127			
INT	Not allocated	Allocated but not used		-32,768...32,767					
DINT	Not allocated	-2,147,483,648...2,147,483,647							
REAL	Not allocated	-3.40282347E ³⁸ ...-1.17549435E ⁻³⁸ (negative values) 0 1.17549435E ⁻³⁸ ...3.40282347E ³⁸ (positive values)							
LINT	Valid Date/Time range is from 1/1/1970 12:00:00 AM coordinated universal time (UTC) to 1/1/3000 12:00:00 AM UTC								

A tag uses additional memory in the controller to store the tag name and symbol, and allocate memory for data.

To manipulate SINT or INT data, the controller converts the values to DINT values, performs the programmed manipulation, and then returns the result to a SINT or INT value. This requires additional memory and execution time when compared to using DINT values for the same operation.

Guidelines for Data Types

Follow these guidelines depending on the data type for your application.

Guideline	Description
Use DINT data types whenever possible	<p>The Logix5000 controllers perform DINT (32 bit) and REAL (32 bit) math operations. DINT data types use less memory and execute faster than other data types. Use the following data type:</p> <ul style="list-style-type: none"> DINT for most numeric values and array indexes. REAL for manipulating floating point, analog values. SINT (8 bit) and INT (16 bit) primarily in user-defined structures or when communicating with an external device that does not support DINT values.


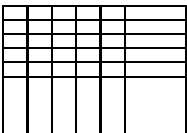
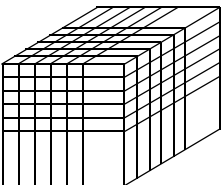
	SINT	INT	DINT	REAL
Memory that is reserved for a standalone tag	4 bytes	4 bytes	4 bytes	4 bytes
Memory that is reserved for data in a user-defined structure	1 byte (8-bit aligned)	2 bytes (16-bit aligned)	4 bytes (32-bit aligned)	4 bytes (32-bit aligned)
Memory that is used to access a tag in an ADD instruction	236 bytes	260 bytes	28 bytes	44 bytes
Execution time on a 1756-L63 controller that is required to perform an ADD instruction	3.31 μs	3.49 μs	0.26 μs	1.45 μs

Group BOOL values into arrays	When you use BOOL values, group them into DINT arrays to best use controller memory and to make the bits accessible via FBC or DDT instructions.
-------------------------------	--

Arrays

An array allocates a contiguous block of memory to store a specific data type as a table of values.

- Tags support arrays in one, two, or three dimensions.
- User-defined structures can contain a single-dimension array as a member of the structure.

This array	Stores data like	For Example				
One dimension		Tag name <i>one_d_array</i>	Type DINT[7]	Dimension 0 7	Dimension 1 --	Dimension 2 --
		Total number of elements = 7 Valid subscript range DINT[a] where a=0...6				
Two dimension		Tag name <i>two_d_array</i>	Type DINT[4,5]	Dimension 0 4	Dimension 1 5	Dimension 2 --
		Total number of elements = 4 * 5 = 20 Valid subscript range DINT[a,b] where a=0...3; b=0...4				
Three dimension		Tag name <i>three_d_array</i>	Type DINT[2,3,4]	Dimension 0 2	Dimension 1 3	Dimension 2 4
		Total number of elements = 2 * 3 * 4 = 24 Valid subscript range DINT[a,b,c] where a=0...1; b=0...2, c=0...3				

The data type you select for an array determines how the contiguous block of memory gets used.

BOOL[96] = 12 bytes

BOOL arrays use 32-bit increments of memory

3	3	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0							
1	0	9	7	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0
6	6	6	6	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	3	3
3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2
9	9	9	9	9	9	8	8	8	8	8	8	8	8	8	8	7	7	7	7	7	7	7	7	7	7	7	6	6	6	6	6
5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4

SINT[10] = 12 bytes of memory (2 bytes unused)

SINT arrays are padded to use any left over bytes

3	2	1	0
7	6	5	4
Unused	Unused	9	8

INT[5] = 12 bytes of memory (2 bytes unused)

INT arrays are padded to use any left over bytes

1	0
3	2
Unused	4

DINT[3] = 12 bytes and REAL[3] = 12 bytes

DINT and REAL arrays use 4-byte increments of memory

0
1
2

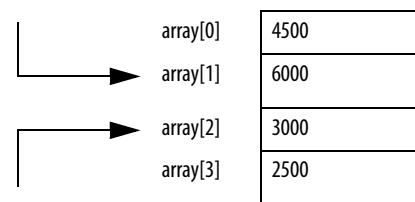
Guidelines for Arrays

Guideline	Description	
You can create arrays of most data types, except for ALARM, AXIS, COORDINATE_SYSTEM, MOTION_GROUP, and MESSAGE data types.	A subscript identifies an individual element within the array. A subscript starts at 0 and extends to the number of elements minus 1 (zero based). <ul style="list-style-type: none"> • Single-dimension arrays take less memory and execute faster than two-dimension or three-dimension arrays. • Direct references to array elements execute faster than indexed references. • An array can be as large as 2 MB. • If you create an array of structures, the memory for each element is allocated based on the structure definition. 	
	Type of Array	Benefit
	Single (1) dimension	<ul style="list-style-type: none"> • Better support by native file instructions • Fully supported in user-defined structures and arrays • Smallest impact (execution time and memory) for indexed references • Can create arrays when programming online
	Double (2) dimension and Triple (3) dimension	<ul style="list-style-type: none"> • Can provide a more accurate data representation for a physical system • Can emulate PLC file/word indirection with a two-dimension array • Can create arrays when programming online
	Considerations	
	<ul style="list-style-type: none"> • Multiple arrays cannot be indirectly referenced like in PLC or SLC processors (such as, N[N7:0]:5) • BOOL arrays are not directly supported by file instructions • Can be changed only when programming offline 	
Nest arrays.	The file instructions offer limited support for arrays. To work with array data, create a user-defined structure with one array as a member of the structure. Then create an array tag by using the user-defined structure as its data type.	
Select the data type of the array based on the data and the instructions that manipulate that data.	While SINT and INT arrays can compact more values into a given memory area, they require additional memory and execution time for each instruction that references the array.	
Limit arrays to 2 MB of data.	The maximum array size is 2 MB. The software displays a warning if you try to create an array that is too large. The software also displays a warning if an array is 1.5...2 MB, even though these sizes are valid.	
Edit arrays online and offline.	You can create arrays when online or offline. However, you can modify only the size or data type of an existing array when offline.	

Indirect Addresses of Arrays

If you want an instruction to access different elements in an array, use a tag in the subscript of the array (an indirect address). By changing the value of the tag, you change the element of the array that your logic references.

When *index* equals 1, *array[index]* points here.



When *index* equals 2, *array[index]* points here.

When you directly reference an element in an array (such as `MyArray[20]`), uses less memory and executes faster than an indirect reference (`MyArray[MyIndex]`). You can also indirectly address bits in a tag (`MyDint.[Index]`).

If you use indirect addresses, use DINT tags because other data types require conversion and execute slower. For each indexed access to data, the controller recalculates the array index. If you access a specific array element multiple times, copy the data out of the array into a fixed tag and use that tag in subsequent logic.

You can also use an expression to specify the index value. For example:
MyArray[10 + MyIndex].

- An expression uses operators to calculate a value.
- The controller computes the result of the expression and uses it as the index.
- These are valid operators.

Operator	Description	Optimal
+	Add	DINT, REAL
-	Subtract/negate	DINT, REAL
*	Multiply	DINT, REAL
/	Divide	DINT, REAL
**	Exponent (x to y)	DINT, REAL
ABS	Absolute value	DINT, REAL
ACS	Arc cosine	REAL
AND	Bitwise AND	DINT
ASN	Arc sine	REAL
ATN	Arc tangent	REAL
COS	Cosine	REAL
DEG	Radians to degrees	DINT, REAL
FRD	BCD to integer	DINT

Operator	Description	Optimal
LN	Natural log	REAL
LOG	Log base 10	REAL
MOD	Modulo divide	DINT, REAL
NOT	Bitwise complement	DINT
OR	Bitwise OR	DINT
RAD	Degrees to radians	DINT, REAL
SIN	Sine	REAL
SQR	Square root	DINT, REAL
TAN	Tangent	REAL
TOD	Integer to BCD	DINT
TRN	Truncate	DINT, REAL
XOR	Bitwise exclusive OR	DINT

Guidelines for Array Indexes

Guideline	Description
Use the SIZE instruction to determine the number of elements in an array.	<p>By determining the number of elements in an array at runtime, you can write reusable code that adjusts itself to meet each instance where it is used.</p> <div style="text-align: center;"> </div> <p>The SIZE instruction returns the number of elements. Arrays are zero-based, so subtract 1 from the result to determine the last element position.</p>
Use immediate values to reference array elements.	Immediate value references to array elements are quicker to process and execute faster than indexed references.
Use DINT tags for array indexes.	DINT tags execute the fastest. SINT, INT, and REAL tags require conversion code that can add additional scan time to an operation.
Avoid using array elements as indexes.	The Logix5000 controller does not directly support the use of an array element as the index to look up a value in another array. To work around this, you can create an alias to the element and then use this as the index. Or copy the element to a base tag and use that base tag as the index.


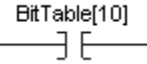
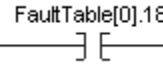

Guidelines for User-defined Structures

Table 1 - UDT Guidelines

Guideline	Description
<p>Group members of the same data type within a structure.</p>	<p>You can create members of most data types, except for ALARM, AXIS, COORDINATE_SYSTEM, MOTION_GROUP, and MESSAGE data types.</p> <p>Place members that use the same data type in sequence.</p> <div style="text-align: center;"> </div> <p>A Logix5000 controller aligns every data type along an 8-bit boundary for SINTs, a 16-bit boundary for INTs, or a 32-bit boundary for DINTs and REALs. BOOLs also align on 8-bit boundaries, but if they are placed next to each other in a user-defined structure, they are mapped so that they share the same byte.</p>
<p>Arrays within structures can only be 1-dimension.</p>	<p>If you include an array as a member, limit the array to one dimension. Multidimension arrays are not permitted in a user-defined structure.</p>
<p>I/O data that is used in structure must be copied into the members.</p>	<p>If you include members that represent I/O devices, you must use logic to copy the data into the members of the structure from the corresponding I/O tags.</p> <p>Make sure that the data type of the structure member matches the I/O data type to avoid data type conversion.</p>
<p>Limit user-defined structures to 500 members.</p>	<p>Logix5000 controllers limit user-defined structures to 500 members. If you need more, consider nesting structures within the main structure.</p>
<p>Limit user-defined structures to 2 MB of data.</p>	<p>The maximum UDT size is 2 MB. The software displays a warning if you try to create an UDT that is too large. The software also displays a warning if the UDT is 1.5...2 MB, even though these sizes are valid.</p>
<p>Limit the size of user-defined structures if they are to be communicated.</p>	<p>Produced and consumed tags are limited to 500 bytes over the backplane and 480 bytes if over a network.</p> <p>RSLinx software can optimize user-defined structures that are less than 480 bytes.</p>
<p>Use the appropriate instruction to load data into a structure.</p>	<p>Load input values into the user-defined structure at the beginning of the program and copy output values from the user-defined structure at the end of the program.</p> <ul style="list-style-type: none"> • Single bit - Examine On (XIC) and Output Energize (OTE) instructions • Contiguous bits - Bit Field Distribute (BTD) instruction • Single value - MOV instruction • Multiple contiguous values -COP/CPS instruction
<p>Use structure descriptions to automatically create tag descriptions.</p>	<p>Enable the Use Pass-through Description workstation option (Tools > Options > Display) to display the descriptions you add to the members of structures for each tag that uses that structure data type.</p>
<p>Online and offline editing.</p>	<p>You can create user-defined structures when online or offline. However, you can modify only an existing structure when offline.</p>

Select a Data Type for Bit Tags

Bits in a Logix5000 controller can exist as: BOOL tags, bits in a BOOL array, bits in elements of a SINT, INT, DINT array, members of a user-defined structure, or as bits in a SINT, INT, DINT member of a user-defined structure.

Tag Type	Description				
BOOL tag MyBit:BOOL 	Each tag accesses a specific bit. Each tag uses 4 bytes. <table border="1"> <thead> <tr> <th>Benefits</th> <th>Considerations</th> </tr> </thead> <tbody> <tr> <td> <ul style="list-style-type: none"> Each bit has a specific tag </td> <td> <ul style="list-style-type: none"> Requires extra bandwidth to communication Uses more memory Cannot use FBC/DDT bit file instructions </td> </tr> </tbody> </table>	Benefits	Considerations	<ul style="list-style-type: none"> Each bit has a specific tag 	<ul style="list-style-type: none"> Requires extra bandwidth to communication Uses more memory Cannot use FBC/DDT bit file instructions
Benefits	Considerations				
<ul style="list-style-type: none"> Each bit has a specific tag 	<ul style="list-style-type: none"> Requires extra bandwidth to communication Uses more memory Cannot use FBC/DDT bit file instructions 				
BOOL array BitTable:BOOL[32] 	A BOOL array combines multiple bits into adjacent words (32-bit words). <table border="1"> <thead> <tr> <th>Benefits</th> <th>Considerations</th> </tr> </thead> <tbody> <tr> <td> <ul style="list-style-type: none"> Consolidates multiple bits into one word Better use of memory Can address all bits in an array by using indirect addressing </td> <td> <ul style="list-style-type: none"> BOOL data type only supported by bit instructions Cannot use file instructions, copy instructions, or DDT/FBC instructions </td> </tr> </tbody> </table>	Benefits	Considerations	<ul style="list-style-type: none"> Consolidates multiple bits into one word Better use of memory Can address all bits in an array by using indirect addressing 	<ul style="list-style-type: none"> BOOL data type only supported by bit instructions Cannot use file instructions, copy instructions, or DDT/FBC instructions
Benefits	Considerations				
<ul style="list-style-type: none"> Consolidates multiple bits into one word Better use of memory Can address all bits in an array by using indirect addressing 	<ul style="list-style-type: none"> BOOL data type only supported by bit instructions Cannot use file instructions, copy instructions, or DDT/FBC instructions 				
DINT array FaultTable:DINT[3] 	A DINT combines multiple bits into adjacent words. <table border="1"> <thead> <tr> <th>Benefits</th> <th>Considerations</th> </tr> </thead> <tbody> <tr> <td> <ul style="list-style-type: none"> Consolidates multiple bits into one word File instructions, copy instructions, and DDT/FBC instructions support DINT arrays Lets you access the bits by element (word) and bit number </td> <td> <ul style="list-style-type: none"> Requires extra planning to indirectly address bits Difficult to address bits in the array by using indirect addressing </td> </tr> </tbody> </table>	Benefits	Considerations	<ul style="list-style-type: none"> Consolidates multiple bits into one word File instructions, copy instructions, and DDT/FBC instructions support DINT arrays Lets you access the bits by element (word) and bit number 	<ul style="list-style-type: none"> Requires extra planning to indirectly address bits Difficult to address bits in the array by using indirect addressing
Benefits	Considerations				
<ul style="list-style-type: none"> Consolidates multiple bits into one word File instructions, copy instructions, and DDT/FBC instructions support DINT arrays Lets you access the bits by element (word) and bit number 	<ul style="list-style-type: none"> Requires extra planning to indirectly address bits Difficult to address bits in the array by using indirect addressing 				
User-defined structure BitStructure Bit1:BOOL Bit2:BOOL Fault:BitStructure 	A user-defined structure combines multiple bits into adjacent, individually named words. <table border="1"> <thead> <tr> <th>Benefits</th> <th>Considerations</th> </tr> </thead> <tbody> <tr> <td> <ul style="list-style-type: none"> Object based Consolidates multiple bits into one word </td> <td> <ul style="list-style-type: none"> Third party MMI/EOI products do not directly support structures. Cannot use FBC/DDT bit file instructions </td> </tr> </tbody> </table>	Benefits	Considerations	<ul style="list-style-type: none"> Object based Consolidates multiple bits into one word 	<ul style="list-style-type: none"> Third party MMI/EOI products do not directly support structures. Cannot use FBC/DDT bit file instructions
Benefits	Considerations				
<ul style="list-style-type: none"> Object based Consolidates multiple bits into one word 	<ul style="list-style-type: none"> Third party MMI/EOI products do not directly support structures. Cannot use FBC/DDT bit file instructions 				

Serial Bit Addresses

The BOOL B data table in the PLC-5 and SLC 500 processors supports two address modes that can address the same bit.

Address Mode	Description
Serial bit In PLC-5 or SLC software, this addressing mode is represented as /Bit	Serial bit addressing references all bits as a contiguous list (array) of bits. For example, if you want to reference the third bit in the second word of a B file, specify B3/18. This method is similar to a BOOL array in a Logix5000 controller where you specify FaultBit[18].
Word bit In PLC-5 or SLC software, this addressing mode is represented as Word/Bit	Word bit addressing identifies a bit within a specific word. For example, B3:1/2 is the same as B3/18 from the serial bit example. This method is similar to accessing the bits of a SINT, INT, DINT array in a Logix5000 controller where you specify FaultTable[1].2.

The Logix5000 controller supports both of these addressing modes, but you cannot use both to reference bits in the same array due to conformance with the IEC 61131-3 standard. Choose the method that best meets your application needs. You can copy data between arrays by using both methods.

You can also use an expression to indirectly reference a bit in a DINT array by using a serialized bit number. For example:

```

Tag
    MyBits : DINT[10]
    BitRef : DINT
EndTag

MOV(34, BitRef)
XIC(MyBits[BitRef / 32].[BitRef AND 31])

```

where:

This expression	Calculates the
[BitRef / 32]	Element in the DINT array
If the tag MyBits is an INT or SINT, the divisor is 16 or 8, respectively.	
[BitRef AND 31]	Bit within the element
If the tag MyBits is an INT or SINT, the mask value is 15 or 7, respectively.	

The Diagnostic Detect (DDT) and File Bit Compare (FBC) instructions provide a bit number as a result of their operation. These instructions are limited to DINT arrays so you can use them to locate the bit number that is returned from the example above.

Guidelines for String Data Types

String data types are structures that hold ASCII characters. The first member of the structure defines the length of the string; the second member is an array that holds the actual ASCII characters.

Name:

Description:

Maximum Characters:

Members: Data Type Size: 516

Name	Data Type	Style	Description
LEN	DINT	Decimal	
DATA	SINT[512]	ASCII	

Guideline	Description
You can create a string data type that is longer or shorter than the default string data type.	The default string data type can contain as many as 82 characters, but you can create custom-length string data types to hold as many characters as needed.
Only some instructions support string data types.	These comparison instructions support string tags: EQU, NEQ, GRT, GEG, LES, LEQ, CMP. These serial port instructions support string tags: ARD, ARL, AWA, AWT. These string-handling instructions support string tags: STOD, DTOS, STOR, RTOS, CONCAT, MID, FIND, DELETE, INSERT, UPPER, LOWER, SIZE. These file instructions support string arrays: FAL, FFL, FFU, LFL, LFU, COP, CPS, FSC.
Use the SIZE instruction to determine the number of characters in a string,	By determining the number of characters in a string at runtime, you can write reusable code that adjusts itself to meet each instance where it is used.
Use the DTOS, RTOS, and CONCAT instructions to embed tag values within a string.	The SLC 500 processor supports the ability to embed a data-table reference address within a string (inline indirection). The SLC 500 AWA and AWT instructions can then look up the data value and place an ASCII representation into the outgoing string. The Logix5000 controller does not directly support this ability. Use the DTOS or RTOS instructions to convert a value to a string and the CONCAT instruction to merge characters with another string.
Set the LEN field to indicate the number characters that are present.	The LEN field in the string structure indicates how many characters are in the string. RSLogix 5000 software and the controller instructions that manipulate strings use the LEN value to determine how many positions in the string DATA array contain valid characters. Both RSLogix 5000 software and the instructions stop processing the DATA array once they reach the LEN value.

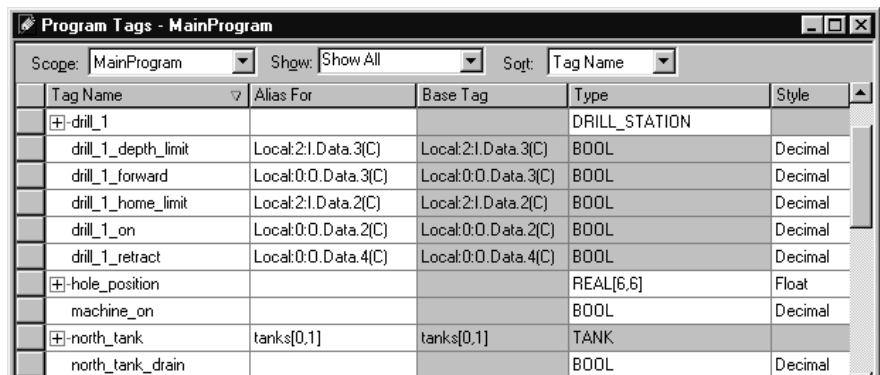
PLC-5/SLC 500 Access of Strings

The ASCII A data table in the PLC-5 and SLC 500 processors uses a string format that is similar to the Logix string data type. The main difference is that the LEN field (length) in a PLC-5/SLC 500 processor is a 16-bit, INT value. The LEN field in a Logix5000 controller is a 32-bit, DINT field. This difference can impact converted logic and data communication. The Logix5000 controller converts the LEN field to the appropriate value and size when a PLC-5/SLC 500 message format is used to read or write a string.

Configure Tags

A tag is a text-based name for an area of the controller's memory where data is stored. Tags are the basic mechanism to allocate memory, reference data from logic, and monitor data.

If you want the tag to	Then choose this type
Store a value for use by logic within the project	Base
Use another name for an existing tag's data (can help simplify long, pre-determined tag names, such as for I/O data or user-defined structures)	Alias
Send (broadcast) data to another controller	Produced
Receive data from another controller	Consumed



Tag Name	Alias For	Base Tag	Type	Style
[-] drill_1			DRILL_STATION	
drill_1_depth_limit	Local:2:1.Data.3(C)	Local:2:1.Data.3(C)	BOOL	Decimal
drill_1_forward	Local:0:0.Data.3(C)	Local:0:0.Data.3(C)	BOOL	Decimal
drill_1_home_limit	Local:2:1.Data.2(C)	Local:2:1.Data.2(C)	BOOL	Decimal
drill_1_on	Local:0:0.Data.2(C)	Local:0:0.Data.2(C)	BOOL	Decimal
drill_1_retract	Local:0:0.Data.4(C)	Local:0:0.Data.4(C)	BOOL	Decimal
[-] hole_position			REAL[6,6]	Float
machine_on			BOOL	Decimal
[-] north_tank	tanks[0,1]	tanks[0,1]	TANK	
north_tank_drain			BOOL	Decimal

For more information on I/O tags, see [Communicate with I/O on page 71](#).

Guidelines for Base Tags

Use the following guidelines for base tags.

Table 2 - Base Tag Guidelines

Guideline	Description
Create standalone atomic tags.	<p>The controller supports pre-defined, standalone tags.</p> <ul style="list-style-type: none"> Atomic tags are listed directly in the Tag Editor and Data Monitor and can easily be found by browsing the alphabetical list. Atomic tags can be created online, but the data type can be only modified offline. <p>Using only atomic tags can impact HMI communication performance as more information must be passed and acted on.</p>
Create user-defined structures	<p>User-defined structures (data types) let you organize your data to match your machine or process.</p> <ul style="list-style-type: none"> One tag contains all data that is related to a specific aspect of your system. This keeps related data together and easy to locate, regardless of its data type. Each piece of data (member) gets a descriptive name. You can use the structure to create multiple tags with the same data layout. User-defined structure can only be modified offline. <p>RSLinx software optimizes user-defined structures more than standalone tags.</p>
Use arrays like files to create a group of similar tags.	<p>An array creates multiple instances of a data type under a common tag name.</p> <ul style="list-style-type: none"> Arrays let you organize a block of tags that use the same data type and perform a similar function. You organize the data in one, two, or three dimensions to match what the data represents. Arrays can be only modified offline. RSLinx software optimizes array data types more than standalone tags. <p>Minimize the use of BOOL arrays. Many array instructions do not operate on BOOL arrays, making it more difficult to initialize and clear an array of BOOL data.</p>

Table 2 - Base Tag Guidelines

Guideline	Description
Take advantage of program-scoped tags.	If you want multiple tags with the same name, define each tag at the program scope (program tags) for a different program. This lets you reuse both logic and tag names in multiple programs. Avoid using the same name for both a controller tag and a program tag. Within a program, you cannot reference a controller tag if a tag of the same name exists as a program tag for that program.
Use mixed case and the underscore characters.	Although tags are not case-sensitive (upper case A is the same as lower case a), mixed case is easier to read. For example, Tank_1 can be easier to read than tank1.
Consider alphabetical order.	RSLogix 5000 software displays tags of the same scope in alphabetical order. To make it easier to monitor related tags, use similar starting characters for tags that you want to keep together. For example, consider using Tank_North and Tank_South rather than North_Tank and South_Tank.
Use leading zeroes (0) when numbers are part of tag names	RSLogix 5000 software uses a simple sort to alphabetize tag names in the Tag Editor and Data Monitor. This means if you have Tag1, Tag2, Tag11, and Tag12, the software displays them in order as Tag1, Tag11, Tag12, and then Tag2. If you want to keep them in numerical order, name them Tag01, Tag02, Tag11, and Tag12.

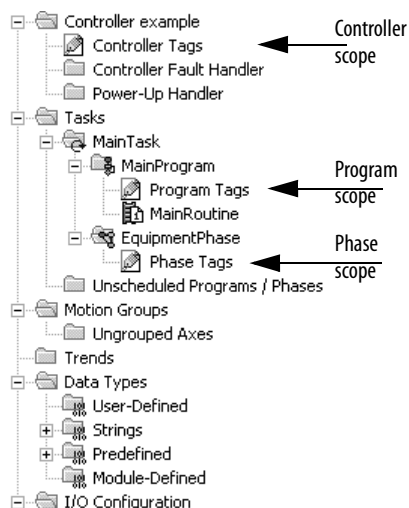
Create Alias Tags

An alias tag lets you create one tag that represents another tag.

- Both tags share the same value as defined by the base tag.
- When the value of a base tag changes, all references (aliases) to the base tag reflect the change.

Guideline	Description
An alias tag references a base tag.	When you assign aliases, avoid: <ul style="list-style-type: none"> • Nesting aliases. • Using multiple aliases to the same tag. On upload, the software decompiles the program and uses the physical memory addresses to determine which tags are referenced in the code. All references to a base tag revert to an alias if one exists. If multiple aliases point to the same tag, RSLogix 5000 software uses the first alias tag (alphabetically) that it finds.
Alias tags do not affect controller execution.	During download, the program is compiled into machine executable code and physical memory addresses. While the existence of an alias requires controller memory to store the name, the program performs the same operation for a reference with an alias or its associated base tag.
Access alias tags from RSLinx software.	Because an alias tag appears as a standalone tag to RSLinx software, an alias tag that references a compound array or structure can require additional communication time. When you reference tags from RSLinx software or other HMI, it can be fastest to reference base tags directly.

Guidelines for Data Scope



Data scope defines where you can access tags. Controller-scoped tags are accessible by all programs. Program-scoped tags are accessible only by the code within a specific program; phase-scoped tags are accessible only by the code within a specific equipment phase.

If you want to	Then assign this scope
Use a tag in multiple programs in the same project	Controller scope (controller tags)
Use a tag in a message (MSG) instruction	
Produce or consume data	
Use motion tags	
Communicate with a PanelView terminal	Program scope (program tags) Phase scope (phase tags)
Reuse the same tag name multiple times for different parts or processes within a controller	
Have multiple programmers work on logic and you want to merge logic into one project	

Isolate portions of a machine or different stations into separate programs or equipment phases and use program-scoped or phase-scoped tags. This lets you do the following:

- Provide isolation between programs and equipment phases
- Prevent tag name collisions
- Improve the ability to reuse code

Guidelines for Tag Names

Use the following guidelines when you name tags.

Guideline	Description
Create descriptive names but keep them short.	Tag names can be from 1...40 characters long. <ul style="list-style-type: none"> • Each character of the tag name uses 1 byte of controller memory, rounded to a 4-byte boundary. • For example, a tag name with 1...4 characters uses 4 bytes. A tag name with 5 characters uses 8 bytes. • Tag names are stored in the controller. • Use structures to reduce the number and size of tags needed. Program upload preserves tag names.
Create a naming convention.	Develop a tag-naming convention on electrical drawings or machine design. For example, Conv1_Full_PE101 combines the sensor function with the photoeye number.
Use correct characters in tag names.	Logix5000 tag names follow the IEC 61131-3 standard. You can use: <ul style="list-style-type: none"> • Letters A through Z. • Numbers 0...9. • Underscore character (_). Tags must start with a letter to avoid confusion with logical expressions. The remaining characters can be any of the supported characters.
Pad names to improve sort order.	RSLogix 5000 software displays tags in alphabetical order. If you use numbers in your tag names, pad the number with leading zeros so the names sort in the proper order. For example, tag names: TS1, TS2, TS3, TS10, TS15, TS20, TS30 display as: TS1, TS10, TS15, TS2, TS20, TS3, and TS30. Pad the numbers with zero so they display as: TS01, TS02, TS03, TS10, TS15, TS20, TS30.

Guidelines for Extended Tag Properties

Use the following guidelines for extended tag properties.

Guideline	Description
Use extended tag properties to define additional information, such as limits, engineering units, or state identifiers, for various components within your controller project.	You can define extended tag properties for these components: <ul style="list-style-type: none"> • Tag • User-defined data type • Add-On Instruction
Some extended tag properties support pass-through for data structures and arrays.	Pass-through behavior is available for descriptions, state identifiers, and engineering units and is configurable in data structures and arrays. Pass-through behavior is not available for limits.
You can read extended properties via logic, but you cannot write to extended properties values in logic.	<ul style="list-style-type: none"> • Extended properties must be used as an input operand. • Alias tags with extended properties cannot be accessed in logic. • Limits can be configured for input and output parameters in Add-On Instructions. However, limit extended properties must not be defined on an InOut parameter of an Add-On Instruction. • Limits cannot be accessed inside Add-On Instruction logic. • If you read an extended property value in logic, it consumes memory equivalent to an equivalent program-scoped tag of that data type. If you do not use them in logic, extended tag properties use no user memory, only extended memory.
If an array tag uses indirect addressing to access limit extended properties in logic, the following conditions apply.	<ul style="list-style-type: none"> • If the array tag has limit extended properties that are configured, the extended properties are applied to any array element that does not explicitly have that particular extended property configured. For example, if the array tag MyArray has Max configured to 100, then any element of the array that does not have Max configured inherits the value of 100 when used in logic. However, it is not visible to you that the value inherited from MyArray is configured in the tag properties. • At least one array element must have specific limit extended property configured for indirectly referenced array logic to verify. For example, if MyArray[x].@Max is being used in logic, at least one array element of MyArray[] must have Max extended property configured if Max is not configured by MyArray. • Under the following circumstances a data type default value is used: <ul style="list-style-type: none"> – Array is accessed programmatically with an indirect reference. – Array tag does not have the extended property configured. – A member of an array does not have the extended property configured.

Tag Descriptions

RSLogix 5000 software searches a tag's origin to locate the first available description. This reduces the number of descriptions you need to enter. This also verifies that tag references display associated descriptions.

Guideline	Description												
Tag descriptions display in RSLogix 5000 software according to the tag's origin.	<table border="1"> <thead> <tr> <th>Type of Tag</th> <th>Description Display in RSLogix 5000 Software</th> </tr> </thead> <tbody> <tr> <td>Atomic</td> <td>For a BOOL, SINT, INT, DINT, or REAL tag, the description that is associated with the tag is the only description available for display.</td> </tr> <tr> <td>Alias</td> <td>First the alias tag description, then the base tag description.</td> </tr> <tr> <td>User-defined structure and Add-On Instruction</td> <td>All members use the description for tag, unless you define a specific description for a member. For example, MyTimer.DN uses the description for MyTimer if there is no description for MyTimer.DN.</td> </tr> <tr> <td>Atomic array</td> <td> <ul style="list-style-type: none"> All references into an array use the description for the array, unless you define a description for an element of the array. For example, MyTable[10] uses the description for MyTable if there is no description for MyTable[10]. All indexed references into an array use the description for the array. For example, MyTable[Index] uses the description for MyTable. </td> </tr> <tr> <td>Structure array</td> <td>All references to a member of a structure in an array default to the array definition, unless you define a description for the structure member of the array. For example, Table[0].Field1 uses the description for Table if there is no description for the specific field.</td> </tr> </tbody> </table>	Type of Tag	Description Display in RSLogix 5000 Software	Atomic	For a BOOL, SINT, INT, DINT, or REAL tag, the description that is associated with the tag is the only description available for display.	Alias	First the alias tag description, then the base tag description.	User-defined structure and Add-On Instruction	All members use the description for tag, unless you define a specific description for a member. For example, MyTimer.DN uses the description for MyTimer if there is no description for MyTimer.DN.	Atomic array	<ul style="list-style-type: none"> All references into an array use the description for the array, unless you define a description for an element of the array. For example, MyTable[10] uses the description for MyTable if there is no description for MyTable[10]. All indexed references into an array use the description for the array. For example, MyTable[Index] uses the description for MyTable. 	Structure array	All references to a member of a structure in an array default to the array definition, unless you define a description for the structure member of the array. For example, Table[0].Field1 uses the description for Table if there is no description for the specific field.
Type of Tag	Description Display in RSLogix 5000 Software												
Atomic	For a BOOL, SINT, INT, DINT, or REAL tag, the description that is associated with the tag is the only description available for display.												
Alias	First the alias tag description, then the base tag description.												
User-defined structure and Add-On Instruction	All members use the description for tag, unless you define a specific description for a member. For example, MyTimer.DN uses the description for MyTimer if there is no description for MyTimer.DN.												
Atomic array	<ul style="list-style-type: none"> All references into an array use the description for the array, unless you define a description for an element of the array. For example, MyTable[10] uses the description for MyTable if there is no description for MyTable[10]. All indexed references into an array use the description for the array. For example, MyTable[Index] uses the description for MyTable. 												
Structure array	All references to a member of a structure in an array default to the array definition, unless you define a description for the structure member of the array. For example, Table[0].Field1 uses the description for Table if there is no description for the specific field.												

For more information, see the Create Tag Descriptions Automatically with User-Defined Data Types White Paper, publication [LOGIX-WP004](#).

Protect Data Access Control at Tag Level

With RSLogix 5000 software, version 18 and later, new tag attributes define access to tag data at runtime.

Tag Attribute	Description
External access	Defines how an external application, such as an HMI, historian, or OPC data server, can access a tag. For arrays, this feature applies to the top level only; for user-defined structure, this feature applies to individual members. Possible values are: <ul style="list-style-type: none"> Read/Write: External applications can both read and modify the tag's value Read Only: External applications can read the tag's value, but not modify it None: External applications can neither read or write the tag's value
Constant	Defines whether a tag value remains constant. Tags with this attribute set cannot be changed programmatically.

Use RSLinx Classic software, version 2.56, and RSLinx Enterprise software, version 5.21 or later, for best results with these tag attributes. Using earlier versions of RSLinx software can result in anomalous behavior from the data servers with the external access options of Read Only and None.

Produced and Consumed Data

Logix5000 controllers support the ability to produce (broadcast) and consume (receive) system-shared tags.

For two controllers to share produced or consumed tags, both controllers must be in the same backplane or attached to the same control network. You cannot bridge produced and consumed tags over two networks.

If there are no other connections, the controller supports these tags.

As a	The controller support
Producer	$(\text{number of produced tags}) \leq 127$
Consumer	$(\text{number of consumed tags}) \leq 250$ (or controller maximum)

The total combined number of consumed and produced tags that a controller supports is:

$$(\text{produced tags}) + (\text{consumed tags}) + (\text{other connections}) \leq 250 \text{ (or controller maximum)}$$

IMPORTANT The actual number of produced and consumed tags that you can configure over ControlNet or EtherNet/IP in a project depends on the connection limits of the communication module through which you produce or consume the tags.

Guidelines for Produced and Consumed Tags

Guideline	Description
You cannot bridge produced and consumed tags over different networks.	For two controllers to share produced or consumed tags, both controllers must be attached to the same network. You can produce and consume tags over ControlNet or EtherNet/IP networks.
Create the tag at controller scope.	You can only produce and consume (share) controller-scoped tags.
Limit the size of the tag to ≤ 500 bytes.	If you transfer a tag with more than 500 bytes, create logic to transfer the data in packets. If you consume a tag over a ControlNet hop, the tag must be ≤ 480 bytes. This is a limitation of the ControlNet network, not the controller.
Combine data that goes to the same controller.	If you are producing several tags for the same controller: <ul style="list-style-type: none"> Group the data into one or more user-defined structures. This uses fewer connections than producing each tag separately. Group the data according to similar update intervals. To conserve network bandwidth, use a greater RPI for less critical data.

Guideline	Description
Use one of these data types: <ul style="list-style-type: none"> • DINT • REAL • Array of DINTs or REALs • User-defined structure 	To share data types other than DINT or REAL, create a user-defined structure to contain the required data. Use the same data type for the produced tag and the corresponding consumed tag or tags.
Use a user-defined structure to produce or consume INT or SINT data.	To produce or consume INT or SINT data, create a user-defined structure with INT or SINT members. The members can be individual INTs or SINTs or the members can be INT or SINT arrays. The resulting user-defined structure can then be produced or consumed.
The data type in the producer and the consumer must match.	The data type for a produced or consumed tag must be the same in both the producer and the consumer.
Produce tags that are based on user-defined structures to non-Logix devices.	The controller produces tags in 32-bit words. For devices that communicate in other word boundaries, such as 16-bit words, the resulting data in the target device can be misaligned. To help avoid misalignment, structure the produced data in a user-defined structure.
Use a programmatic handshake to help ensure data is exchanged.	Produced tags continually transmit based on the RPI, so it can be difficult to know when new data arrives. You can set a bit or increment a counter that is embedded in the produced tag to identify to the consumer that new data is present. You can also provide a return handshake via a reverse produced/consumed tag, so that the original producer knows that the consumer received and processed the tag.
Use a CPS instruction to buffer produced and consumed data.	Use the CPS instruction to copy the data to the outgoing tag on the producer side. Then use another CPS instruction to copy the data into a buffer tag on the consumer side. The CPS instructions provide data integrity for data structures greater than 32 bits. Important: The controller inhibits all interrupts while it executes a CPS instruction.
Use unicast EtherNet/IP communication to reduce broadcast network traffic.	To reduce bandwidth use and preserve network integrity, some facilities block multicast Ethernet packets. With RSLogix 5000 software, version 16, you can configure a produced and consumed tag to use multicast or unicast connections. Unicast connections help with the following: <ul style="list-style-type: none"> • Reduce network bandwidth • Simplify Ethernet switch configuration

Guidelines to Specify an RPI Rate for Produced and Consumed Tags

When configuring produced and consumed tags, you specify a requested packet interval (RPI) rate. The RPI value is the rate at which the controller attempts to communicate with the module.

Guideline	Description
Make sure that the RPI is equal to or greater than the NUT.	You use RSNetWorx™ for ControlNet™ software to select the network update time (NUT) and the software schedules the network connections. RSNetWorx™ software cannot schedule a ControlNet network if a module and/or produced/consumed tag on the network has an RPI that is faster than the network update time.
The smallest (fastest) consumer RPI determines the RPI for the produced tag.	If multiple consumers request the same tag, the smallest (fastest) request determines the rate at which the tag is produced for all consumers.

Guidelines to Manage Connections for Produced and Consumed Tags

Guideline	Description
Minimize the use of produced and consumed tags.	To reduce network traffic, minimize the size of produced and consumed tags. Also, minimize the use of produced and consumed tags to high-speed, deterministic data, such as interlocks.
Use arrays or user-defined structures.	When sending multiple tags to the same controller, use an array or user-defined structure to consolidate the data. The byte limit of ≤ 500 bytes per produced and consumed tag still applies.
Configure the number of consumers accurately.	Make sure the number of consumers that are configured for a produced tag is the actual number of controllers that consumes the tag. If you set the number higher than the actual number of controllers, you unnecessarily use up connections. The default is two consumers per produced tag.
Multiple produced/consumed connections are linked.	If there are multiple produced and consumed connections between two controllers and one connection fails, all produced and consumed connections fail. Consider combining all produced and consumed data into one structure or array so that you only need one connection between the controllers.

Configure an Event Task Based on a Consumed Tag

An event task executes automatically based on a preconfigured event occurring. One such event can be the arrival of a consumed tag.

- Only one consumed tag can trigger a specific event task.
- Use an IoT instruction in the producing controller to signal the production of new data.
- When a consumed tag triggers an event task, the event task waits for all data to arrive before the event task executes.

For information on configuring an event task, see [Configure an Event Task on page 30](#).

Compare Messages and Produced/Consumed Tags

Method	Benefits	Considerations
Read/Write Message	<ul style="list-style-type: none"> • Programmatically initiated • Communication and network resources that are only used when needed • Support automatic fragmentation and reassembly of large data packets, up to as many as 32,767 elements • Some connections can be cached to improve retransmission time • Generic CIP message useful for third-party devices 	<ul style="list-style-type: none"> • Delay can occur if resources are not available when needed • MSG instruction and processing can impact controller scan (system overhead timeslice) • Data arrives asynchronous to program scan (use a programmatic handshake or an UID/UIE instruction pair to reduce impact, no event task support) • Can add additional messages online in Run mode.
Produced/Consumed Tag	<ul style="list-style-type: none"> • Configured once and sent automatically based on requested packet interval (RPI) • Multiple consumers can simultaneously receive the same data from a produced tag • Can trigger an event task when consumed data arrives • ControlNet resources are reserved up front • Does not affect the scan of the controller 	<ul style="list-style-type: none"> • Support limited to Logix5000 and PLC-5 controllers, and the 1784-KTCS I/O Linx and select third-party devices • Limited to 500 bytes over the backplane and 480 bytes over a network • Must be scheduled when using ControlNet • Data arrives asynchronous to program scan (use a programmatic handshake or CPS instruction and event tasks to synchronize) • Connection status must be obtained separately • With RSLogix 5000 version 17 and later, you can configure status information for a produced/consumed tag • On an EtherNet/IP network, you can configure produced/consumed tags to use multicast or unicast connections. • Cannot create additional produced/consumed tags online in Run mode.

Communicate with I/O

In Logix5000 controllers, I/O values update at a period, requested packet interval (RPI), which you configure via Module Property dialog in the I/O configuration folder of the project. The values update asynchronously to the execution of logic.

The module sends input values to the controller at the specified RPI. Because this transfer is asynchronous to the execution of logic, an I/O value in the controller can change in the middle of a scan.

Buffer I/O Data

If you reference an I/O tag multiple times, and the application could be impacted if the value changes during a program scan, you must copy the I/O value into a buffer tag before the first reference of that tag in your code. In your code, reference the buffer tag rather than the I/O tag.

IMPORTANT Use the synchronous copy (CPS) instruction to buffer I/O data. While the CPS instruction copies data, no I/O updates or other tasks can change the data. Tasks that attempt to interrupt a CPS instruction are delayed until the instruction is done. Overuse of the CPS instruction can impact controller performance by keeping all other tasks from executing.

Buffer I/O data to do the following:

- Prevent an input or output value from changing during the execution of a program. (I/O updates asynchronous to the execution of logic.)
- Copy an input or output tag to a member of a structure or element of an array.
- Prevent produced or consumed data from changing during the execution of a program.
- Make sure all produced and consumed data arrives or is sent as a group (not mixed from multiple transfers)
- Only use the CPS instruction if the I/O data that you want to buffer is greater than 32 bits (or 4 bytes) in size

Overuse of the CPS instruction can greatly impact controller performance.

If you have a user-defined structure with members that represent I/O devices, you must use logic to copy the data into the members of the structure from the corresponding I/O tags.

Guidelines to Specify an RPI Rate for I/O Modules

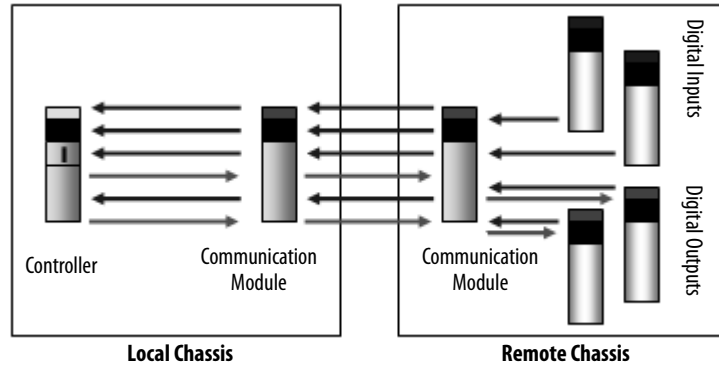
Configure an RPI rate per module (ControlLogix and SoftLogix) or an RPI rate per controller (CompactLogix). The RPI value is the rate at which the controller attempts to communicate with the module.

Guideline	Description
Specify an RPI at 50% of the rate you actually need.	Setting the RPI faster (specifying a smaller number) than what your application needs wastes network resources, such as ControlNet schedule bandwidth, network processing time, and CPU processing time. For example, if you need information every 80 ms, set the RPI at 40 ms. The data is asynchronous to the controller scan, so you sample data twice as often (but no faster) than you need it to make sure that you have the most current data.
Group devices with similar performance needs onto the same module.	By grouping devices with similar performance needs on the same module, you consolidate data transmission to one module rather than multiple modules. This conserves network bandwidth.
Set the ControlNet network update time (NUT) equal to or less than the fastest RPI.	When configuring a ControlNet network, set the network update time (NUT) equal to or less than the fastest RPI of the I/O modules and produced/consumed tags in the system. For example, if your fastest RPI is 10 ms, set the NUT to 5 ms for more flexibility in scheduling the network.
In an ControlNet system, use even multiples of the NUT for the RPI value.	Set the RPI to a binary multiple of the NUT. For example, if the NUT is 10 ms, select an RPI such as 10, 20, 40, 80, or 160 ms.
In a ControlNet system, isolate I/O communication.	If you use unscheduled ControlNet communication or want to be able to add ControlNet I/O at runtime (see page 83), dedicate one ControlNet network to I/O communication only. On the dedicated I/O network, make sure that there is the following: <ul style="list-style-type: none"> • No HMI traffic • No MSG traffic • No programming workstations • No peer-to-peer interlocking in multi-processor system architectures
In an EtherNet/IP system, module change of state is limited to 1/4 of the RPI.	If you configure change of state communication for a module in a remote chassis that is connected via an EtherNet/IP network, the module can send data only as fast as the module RPI. Initially, the module sends its data immediately. However, when an input changes, the module data is held at the adapter until 1/4 of the RPI is reached to avoid overloading the EtherNet/IP network with the module communication.
Data transmission depends on the controller.	The type of controller determines the data transmission rate. <ul style="list-style-type: none"> • ControlLogix and SoftLogix controllers transmit data at the RPI you configure for the module. • CompactLogix controllers transmit data at powers of 2 ms (such as 2, 4, 8, 16, 64, or 128). For example, if you specify an RPI of 100 ms, the data actually transfers at 64 ms.

Communication Formats for I/O Modules

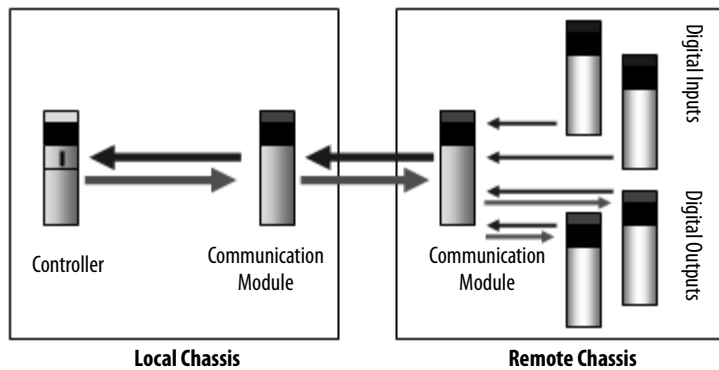
The communication format determines whether the controller connects to the I/O module via a direct or a rack-optimized connection. The communication format also determines the type and quantity of information that the module provides or uses.

Direct connection Each module passes its data to/from the controller individually. Communication modules bridge data across networks.



Benefits	Considerations
<ul style="list-style-type: none"> • Each module can determine its own rate (RPI) • More data can be sent per module, such as diagnostic and analog data • Supports event task communication 	<ul style="list-style-type: none"> • Requires additional connections and network resources • This is the only method supported in the local chassis • I/O data presented as individual tags

Rack-optimized connection The communication module in a remote chassis consolidates data from multiple modules into a single packet and transmits that packet as a single connection to the controller.

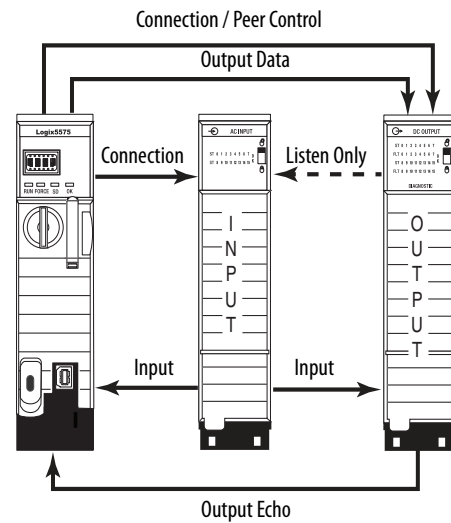


Benefits	Considerations
<ul style="list-style-type: none"> • One connection can service a full chassis of digital modules • Reduces network resources and loading 	<ul style="list-style-type: none"> • All modules are sent at the same rate • Unused slots are still communicated • Still need a direct connection for analog and diagnostic data • Limited to remote chassis • I/O data presented as arrays with alias tags for each module

The rack-optimized format limits data to one 32-bit input word per module in a chassis. If you place a diagnostic module in a chassis, the rack-optimized format eliminates the value that the diagnostic module offers. In this case, it's better to use a direct connection so that the diagnostic information from the module is passed to the controller.

Peer control Output modules let peer ownership of input modules to consume input data to directly control outputs without requiring controller processing. The 1756-IB16IF and 1756-IB16IFC modules can be listened to presuming the output module knows the input data layout and connection information. The configuration from the controller defines how the peer input data is mapped to the output modules. The controller can use the other digital points on the module that are not peer-owned as conventional outputs.

The controller can also use the output data it normally sends to the module with consumed inputs, letting ‘gate-type’ features enabled by controller logic selectively letting application of the consumed peer input data.



Benefits	Considerations
<ul style="list-style-type: none"> • Faster response time because the controller scan time is removed from the equation. Data is sent directly to the output module from the input module. • Increases controller performance by reducing the need for event tasks to close loops quickly. • Each input module has an AND and OR bit mask that defines the logic that is applied to each input module. 	<ul style="list-style-type: none"> • You must program the controller for proper relationship with the output modules. • The peer output module must be in the same chassis as the input module to maximize response time.

Electronic Keying

Electronic Keying reduces the possibility that you use the wrong device in a control system. It compares the device that is defined in your project to the installed device. If keying fails, a fault occurs. These attributes are compared.

Attribute	Description
Vendor	The device manufacturer.
Device Type	The general type of the product, for example, digital I/O module.
Product Code	The specific type of the product. The Product Code maps to a catalog number.
Major Revision	A number that represents the functional capabilities of a device.
Minor Revision	A number that represents behavior changes in the device.

The following Electronic Keying options are available.

Keying Option	Description
Compatible Module	<p>Lets the installed device accept the key of the device that is defined in the project when the installed device can emulate the defined device. With Compatible Module, you can typically replace a device with another device that has the following characteristics:</p> <ul style="list-style-type: none"> • Same catalog number • Same or higher Major Revision • Minor Revision as follows: <ul style="list-style-type: none"> – If the Major Revision is the same, the Minor Revision must be the same or higher. – If the Major Revision is higher, the Minor Revision can be any number.
Disable Keying	<p>Indicates that the keying attributes are not considered when attempting to communicate with a device. With Disable Keying, communication can occur with a device other than the type specified in the project.</p> <p>ATTENTION: Be cautious when using Disable Keying; if used incorrectly, this option can lead to personal injury or death, property damage, or economic loss.</p> <p>We strongly recommend that you do not use Disable Keying.</p> <p>If you use Disable Keying, you must take full responsibility for understanding whether the device being used can fulfill the functional requirements of the application.</p>
Exact Match	Indicates that all keying attributes must match to establish communication. If any attribute does not match precisely, communication with the device does not occur.

Carefully consider the implications of each keying option when selecting one.

IMPORTANT When you change Electronic Keying parameters online, it interrupts connections to the device and any devices that are connected through the device. Connections from other controllers can also be broken.

If an I/O connection to a device is interrupted, the result can be a loss of data.

More Information

For more detailed information on Electronic Keying, see Electronic Keying in Logix5000 Control Systems Application Technique, publication [LOGIX-AT001](#).

Guidelines to Manage I/O Connections

Use the following guidelines to administer your I/O modules.

Table 3 - I/O Connection Guidelines

Guideline	Description
The type of I/O module can determine the type of connection.	Analog modules always use direct connections, except for 1771 analog modules that use messaging. Digital modules can use direct or rack-optimized connections. Communication formats that include optimization in the title are rack-optimized connections; all other connection options are direct connections.
Select the communication format for a remote adapter based on the remote I/O modules.	Select one of these formats for a remote adapter.
Select	If
None	The remote chassis contains only analog modules, diagnostic digital modules, fused output modules, or communication modules. On a ControlNet network, use None to add a new chassis to the network while the controller is running.
Rack-Optimized	The remote chassis only contains standard, digital input, and output modules (no diagnostic modules or fused output modules). For a ControlNet network at runtime (controller is online), you can add new digital modules to an existing rack-optimized connection, but new rack-optimized connections can only be added when offline. An EtherNet/IP network supports new rack optimized connections both offline and at runtime (online). For more information, see page 83 .
Listen Only Rack-Optimized	You want to receive I/O module and chassis slot information from a rack-optimized remote chassis that is owned by another controller. The runtime capability for listen only rack-optimized connections is the same as for rack-optimized connections.
Use rack-optimized connections to conserve connections	If you are trying to limit the number of controller and network connections, rack-optimized connections can help.
In some cases, all direct connections work best.	For a remote adapter that is configured for rack-optimized connections, there is always data that is sent for each slot in the chassis, even if a slot is empty or contains a direct connection module. There are 12 bytes of data that is transferred for rack-optimized overhead between the controller and the remote adapter. In addition, the remote adapter sends 8 bytes per slot to the controller; the controller sends 4 bytes per slot to the remote adapter. For a few digital modules in a large chassis, it can be better to use direct connections because transferring the full chassis information can require more system bandwidth than direct connections to a few modules.
Example	Description
Remote 17-slot chassis Slot 0: 1756-CNBR/D Slots 1...15: analog modules Slot 16: standard digital module	Option 1: Select Rack Optimization as the communication format for the remote adapter. This example uses 16 controller connections (15 for analog modules and 1 for the rack-optimized connection). This example also transfers: <ul style="list-style-type: none"> • 12 bytes for rack-optimized overhead. • 12 bytes for the digital module. • 12 bytes for each of the 15 analog modules, for a total of 180 bytes. Option 2: Select None as the communication format for the remote adapter. This example also uses 16 controller connections (1 direct connection to each I/O module). There is no rack-optimized overhead data to transfer. Recommendation: Option 2 is recommended because it avoids unnecessary network traffic, and thus improves network performance.
Remote 17-slot chassis Slot 0: 1756-CNBR/D Slots 1...8: analog modules Slots 9...16: digital modules	Option 1: Select Rack Optimization as the communication format for the remote adapter. This example uses nine controller connections (eight for analog modules and one for the rack-optimized connection). This example also transfers: <ul style="list-style-type: none"> • 12 bytes for rack-optimized overhead. • 12 bytes for each of the 8 digital modules, for a total of bytes 96 bytes. • 12 bytes for each of the 8 analog modules, for a total of 96 bytes. Option 2: Select Rack Optimization for the communication format of the remote adapter. This example uses 16 controller connections (1 direct connection to each I/O module). There is no rack-optimized overhead data to transfer. Recommendation: The best option for this example depends on the type of digital I/O modules in the system and other controller connections. If the total system has many analog modules, diagnostic modules, fused output modules, or produced/consumed tags, select Option 1 to conserve controller connections. If there are plenty of controller connections available, select Option 2 to reduce unnecessary network traffic.

Control 1771 I/O Modules

The Logix5000 controllers support the following:

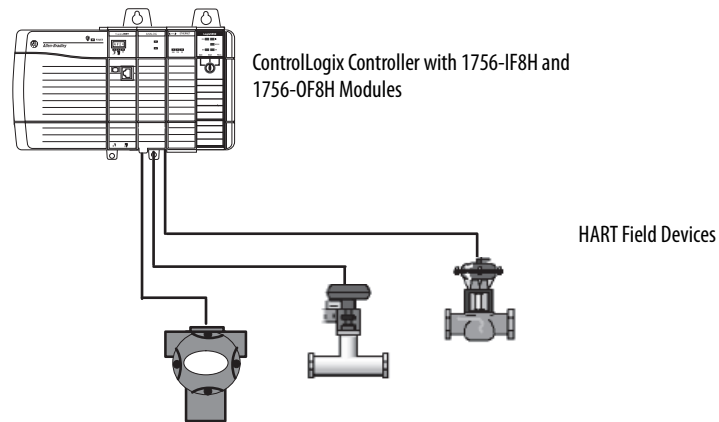
- Remote I/O communication to 1771 digital and analog I/O modules
- ControlNet communication to 1771 digital I/O modules
- Block transfer message instructions via a remote I/O or ControlNet network to 1771 analog and intelligent I/O modules

Guideline	Description
Distribute 1771 analog I/O modules.	Spread 1771 I/O analog I/O modules across multiple chassis to reduce the number of block transfers one 1771-ACN15, 1771-ACNR15, or 1771-ASB adapter manages. Isolate different 1771 chassis on different networks to diversify the communication so no single network communication module has to manage all block transfer messages.
For block transfers over a ControlNet network, increase the amount of ControlNet unscheduled bandwidth.	The traffic load of scheduled communication determines the amount of time available for unscheduled communication. <div style="text-align: center; margin: 10px 0;"> </div> Increase the controller system overhead to allocate more CPU time to message and block transfer processing.
Program block transfers.	Unscheduled data is limited to 510 bytes/node per ControlNet NUT. The 1756-CNB is limited to 128 words per transfer. If needed, data is sent in multiple packets. The data transfer occurs asynchronous to the program scan. See page 101 for more information on block transfers.

Communicate with HART Devices

HART (Highway Addressable Remote Transmitter) is an open protocol that is designed to connect analog devices in industrial process-measurement applications. The protocol uses the standard 4...20 mA current loop that is widely used for such measurements.

The 1756-IFxH and 1756-OFxH modules offer analog and HART connectivity in one module. You can place modules local to the controller or remote over ControlNet or EtherNet/IP networks. You do not need external hardware to access the HART signal.



Guideline	Description
Enable HART support on only those channels that need the support.	All channels share the HART modem, so HART response time is better if you enable only the needed HART channels.
The update rate for the HART part of a tag is slower than for the analog part.	The update rate varies, depending on HART network traffic. If all eight channels have HART enabled, update rates are in the range of 10 s. Be sure to consider this response time in your control strategy. Also, check the data quality indications that are provided with the HART data.
The Device Variable Status (PVStatus, SVStatus, TVStatus, FVStatus) is a relatively new feature in HART systems.	If your HART device does not support Device Variable Status, the 1756-IF8H and 1756-OF8H module synthesizes a status value that is based on the communication status with the HART field device. The Dynamic Variables do not update as fast as the Analog Signal. The actual rate depends on: <ul style="list-style-type: none"> • The number of channels that are configured for HART. • The number of Pass Through commands. • The presence of handheld communicators or other secondary masters. • The response speed of the field device.
The 1756 HART modules support asset management software.	HART must be enabled before any asset management access is possible, including scanning for multiplexors. RSLinx Professional software, RSLinx Gateway software, and RSLinx OEM software let asset management software communicate through networks and the 1756 backplane. Endress+Hauser FieldCare asset management software is a Field Device Tool (FDT) frame application. The frame application runs the Device Type Manager (DTM) files. The DTM files are executable files that are provided by control and device vendors. There are communication DTMs and device DTMs. Rockwell Automation provides one communication DTM for RSLinx software and the 1756 backplane and two other communication DTMs for the 1756 HART modules. Companies like Endress+Hauser provide device DTMs for their instruments and valves. The device DTMs provide visualization of the parameters that are used to configure, monitor, and maintain the devices.

Communicate with FOUNDATION Fieldbus Devices

FOUNDATION Fieldbus is a communication network that is created by the Fieldbus Foundation. It is a protocol that is designed for distributed control of process control applications.

If your application bridges from	Select	Description
EtherNet/IP	1757-FFLD2 1757-FFLD4	The 1757-FFLDx linking device bridges from an Ethernet network to either two or four H1 ports.
ControlNet	1757-FFLDC2 1757-FFLDC4	The 1757-FFLDCx linking device bridges from a ControlNet network to either two or four H1 ports. The 1757-FFLDCx is compatible with ControlLogix redundancy and supports redundant ControlNet media.

Figure 1 - Example Configuration - EtherNet/IP Network to FOUNDATION Fieldbus Network

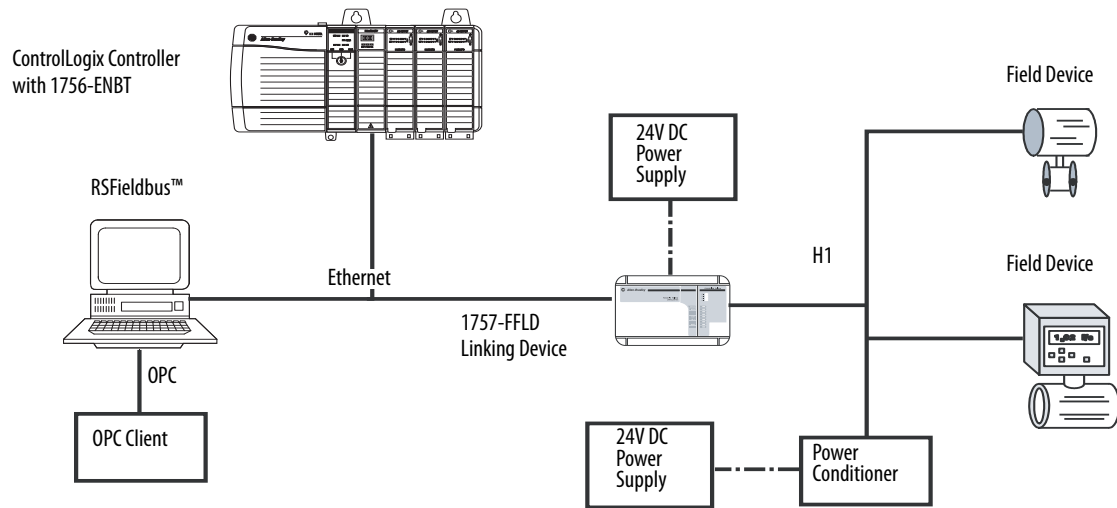


Figure 2 - Example Configuration - ControlNet Network to FOUNDATION Fieldbus Network

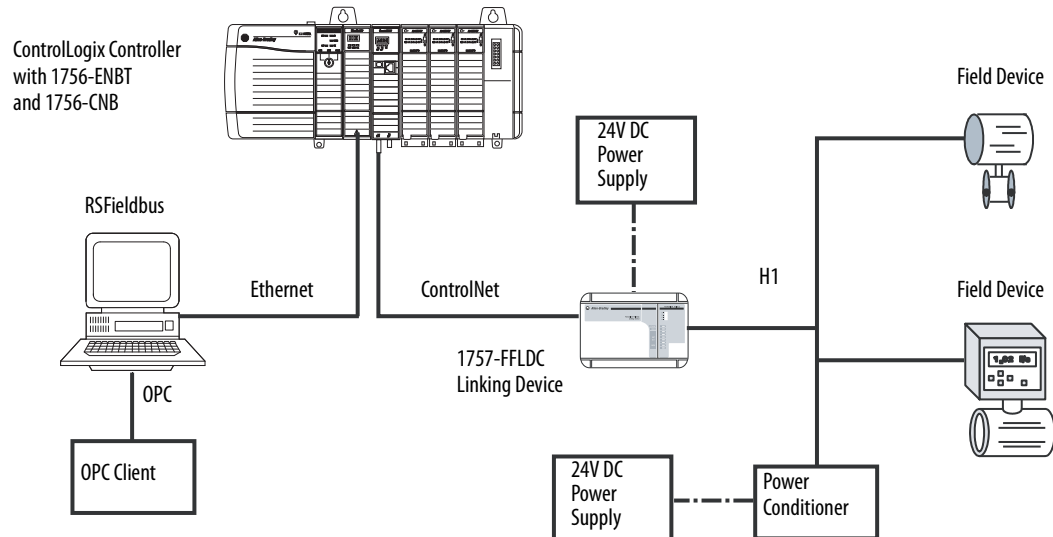


Table 4 - Linking Device Guidelines

Guideline	Description
Use either the EtherNet/IP linking device or the ControlNet linking device with an HSE server.	Rockwell Automation does not support 1757-FFLDC and 1757-FFLD linking devices communicating with the same HSE server in the same RSFieldbus project.
Each linking device supports 16 Logix blocks.	Each Logix block supports eight digital inputs, eight digital outputs, eight analog inputs, and eight analog outputs. Each Logix block uses one CIP connection.
The connections that are required for a linking device depend on the number of Logix blocks.	Each linking device uses these connections: <ul style="list-style-type: none"> • Two connections to the network communication module • One connection for each Logix block
The type of device affects the maximum number of FOUNDATION Fieldbus devices per H1 segment.	Each linking device supports two or four H1 segments, with 8...10 instruments (16 maximum) per each H1 segment.
Do not exceed the maximum number of virtual communication relationships (VCRs) on each H1 segment.	A VCR is a channel that provides for the transfer of data between FOUNDATION Fieldbus devices. The number of VCRs required to send data or receive data depends on the device and type of data. Each parameter that you pass to or from the Logix5000 controller uses a VCR. Some devices, such as valves, use more VCRs than transmitters. As of firmware revision 2.1 and later, the linking device supports a maximum of 64 publisher and 64 subscriber VCRs for each H1 segment. Earlier firmware revisions support a maximum of 16 publisher and 16 subscriber VCRs for each H1 segment.
Make sure that you have the correct device description (DD) for each linking device.	DDs are like EDS files for DeviceNet devices. You can find DDs on vendor/organization websites or on media that ships with the device. A host with DD services can interoperate with all parameters defined in the DD for a field device.
Use the right wiring and connection products.	Always use a tree or modified tree topology. Never daisy chain devices. Noise is the most frequent problem, due to: <ul style="list-style-type: none"> • Wrong wiring • Improper grounding • Bad connectors
To get the best implementation, understand the details of a FOUNDATION Fieldbus system.	See these references: <ul style="list-style-type: none"> • FOUNDATION Fieldbus System User Manual, publication 1757-UM012 • FOUNDATION Fieldbus Design Considerations Reference Manual, publication RSFBUS-RM001 • FOUNDATION Fieldbus Technical Overview at http://www.fieldbus.org • Relcom Wiring Guide at http://www.relcominc.com • Pepperl-Fuchs Fieldbus Wiring and Installation Guide

Create Tags for I/O Data

Each I/O tag is automatically created when you configure the I/O module through the programming software. Each tag name follows this format:

Location:SlotNumber:Type.MemberName.SubMemberName.Bit

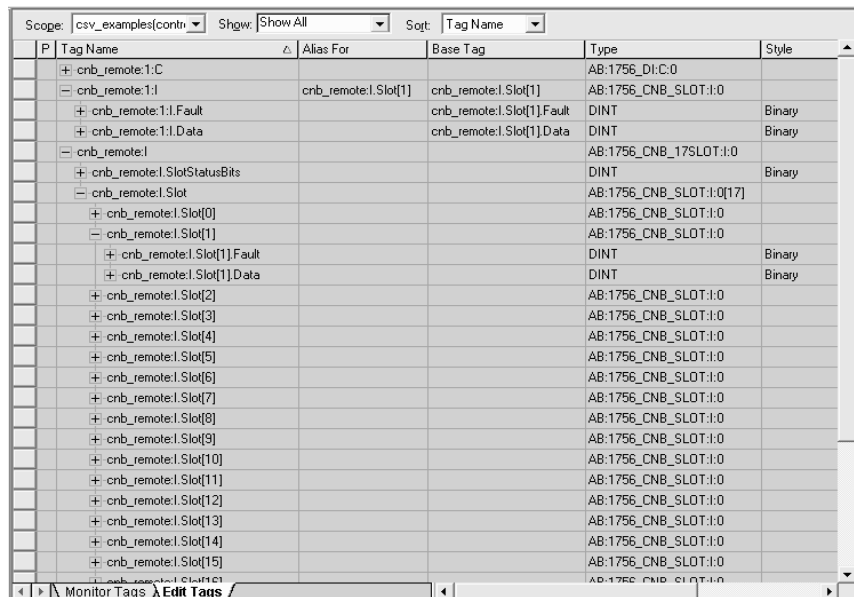
This address variable	Is
Location	Identifies network location LOCAL = local chassis or DIN rail ADAPTER_NAME = identifies remote adapter or bridge
SlotNumber	Slot number of I/O module in its chassis
Type	Type of data: I = input C = configuration O = output S = status
MemberName	Specific data from the I/O module, such as Data and Fault; depends on the module
SubMemberName	Specific data that is related to a MemberName
Bit (optional)	Specific point on the I/O module; depends on the size of the I/O module (0...31 for a 32-point module)

If you configure a rack-optimized connection, the software creates a rack-object tag for the remote communication module. You can reference the rack-optimized I/O module individually, or by its element within the rack-object tag.

For example, a remote ControlNet communication module (remote_cnb) has an I/O module in slot 1.

The individual tag that is created for the I/O module in remote slot 1.

The entry in the rack-object tag for the remote communication module that identifies the I/O module in remote slot 1.



Controller Ownership

When you choose a communication format, you have to choose whether to establish an owner or listen-only relationship with the module.

Mode	Description
Owner	The owner controller writes configuration data and can establish a connection to the module.
Listen-only	A controller that uses a listen-only connection only monitors the module. It does not write configuration data and can only maintain a connection to the I/O module when the owner controller is actively controlling the I/O module.

There is a noted difference in the ownership of input modules versus the ownership of output modules.

Controlling	This Ownership	Description
Input modules	Owner	An input module is configured by a controller that establishes a connection as an owner. This configuring controller is the first controller to establish an owner connection. Once an input module has been configured (and owned by a controller), other controllers can establish owner connections to that module. This lets additional owners to continue to receive multicast data if the original owner controller breaks its connection to the module. All other additional owners must have the identical configuration data and identical communication format that the original owner controller has, otherwise the connection attempt is rejected.
	Listen-only	Once an input module has been configured (and owned by a controller), other controllers can establish a listen-only connection to that module. These controllers can receive multicast data while another controller owns the module. If all owner controllers break their connections to the input module, all controllers with listen-only connections no longer receive multicast data.
Output modules	Owner	An output module is configured by a controller that establishes a connection as an owner. Only one owner connection can be connected to an output module. If another controller attempts to establish an owner connection, the connection attempt is rejected.
	Listen-only	Once an output module has been configured (and owned by one controller), other controllers can establish listen-only connections to that module. These controllers can receive multicast data while another controller owns the module. If the owner controller breaks its connection to the output module, all controllers with listen-only connections no longer receive multicast data.

Runtime/Online Addition of Modules

You can add modules when the controller is in Run mode.

RSLogix 5000 Software	Support for Online Addition of Modules
Version 15 and later	Add 1756 I/O modules to the local chassis, remotely via the unscheduled portion of a ControlNet network, and remotely via an EtherNet/IP network.
Version 17 and later	Add a 1757-FFLDC module remotely via the unscheduled portion of a ControlNet network.

Network	Considerations
ControlNet network	<p>You can use:</p> <ul style="list-style-type: none"> 1756-CN2, 1756-CN2R, 1756-CN2RXT any series modules. 1756-CNB, 1756-CNBR series D or later communication modules. <p>Digital I/O modules can be added as rack-optimized connections if the parent module is already configured with rack-optimized connections. While you can add a new digital I/O module to an existing rack-optimized connection, you cannot add rack-optimized connections while online. Digital I/O modules can also be added as direct connections.</p> <p>Analog I/O modules can be added only as direct connections.</p> <p>Disable the Change of State (COS) feature on digital input modules because it can cause inputs to be sent more quickly than the RPI.</p> <p>If you plan to add large amounts of I/O to the ControlNet network, dedicate one ControlNet network for I/O. For the dedicated ControlNet network, verify that there is little or no:</p> <ul style="list-style-type: none"> HMI traffic. MSG traffic. Programming workstations. <p>If the module has a Real Time Sample (RTS), disable it or set to a rate that is greater than the RPI.</p> <p>Considerations for 1756-CN2, 1756-CN2R, 1756-CN2RXT Modules</p> <p>You can add I/O modules until you reach these limits:</p> <ul style="list-style-type: none"> 80% of CPU utilization of the 1756-CN2, 1756-CN2R, or 1756-CN2RXT communication module. Less than 400,000 unscheduled bytes per second are displayed in RSNetWorx™ for ControlNet software after the network has been scheduled. <p>Considerations for 1756-CNB, 1756-CNBR Modules</p> <p>Requested Packet Intervals (RPIs) faster than 25 ms for unscheduled modules can overload the 1756-CNB or 1756-CNBR communication module. To avoid the overload, make these considerations:</p> <ul style="list-style-type: none"> Use a NUT of 10 ms or more. Keep the SMAX and UMAX values as small as possible. <p>You can add I/O modules until you reach these limits:</p> <ul style="list-style-type: none"> 75% of CPU utilization of the 1756-CNB or 1756-CNBR communication module. Plan for a CPU-use increase of 1...4% of the 1756-CNB or 1756-CNBR module for each I/O module you add, depending on RPI. 48 connections on the 1756-CNB or 1756-CNBR communication module. Less than 400,000 unscheduled bytes per second are displayed in RSNetWorx for ControlNet software after the network has been scheduled.
EtherNet/IP network	<p>The EtherNet/IP I/O modules that you add at runtime can be:</p> <ul style="list-style-type: none"> Added to existing rack-optimized connections Added to new rack-optimized connections Added as direct connections (you can create rack-optimized connections when adding EtherNet/IP I/O modules at runtime) <p>You can add I/O modules until you reach the limits of the communication module.</p>

1756-EN2TR, 1756-EN3TR Module	1756-EN2T, 1756-EN2TXT, 1756-EN2F Module	1756-ENBT Module	1756-ENET/B Module
20,000 pps	10,000 pps	5000 pps	900 pps
128 TCP connections	128 TCP connections	64 TCP connections	64 TCP connections
256 CIP connected messages	256 CIP connected messages	128 CIP connected messages	160 CIP connected messages

Add Modules at Runtime/Online

Module Type and Connection Method	In Local Chassis		Remote via a ControlNet Network				Remote via an EtherNet/IP Network		Configure Hold Last Output State
	Offline	Runtime ⁽¹⁾	Offline		Runtime ⁽¹⁾		Offline	Runtime ⁽¹⁾	Offline only
			Scheduled	Unscheduled	Scheduled	Unscheduled			
Motion - direct	Yes	No	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Digital - direct	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes - 1756 I/O digital output modules
Digital - rack-optimized	N/A	N/A	Yes	No	Yes	No	Yes	Yes	Yes - 1756 I/O digital output modules
Analog - direct	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes
Generic third party - direct	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	N/A
1756-DNB	Yes	No	Yes	No	No	No	Yes	Yes	N/A
1756-DHRIO	Yes	No	Yes	No	No	No	Yes	Yes	N/A
1756-CNx - no connection	Yes	Yes	Yes	Yes	No	Yes	N/A	N/A	N/A
1756-CNx - rack-optimized	N/A	N/A	Yes	N/A	N/A	N/A	N/A	N/A	N/A
Generic ControlNet third party - direct	N/A	N/A	Yes	Yes	No	Yes	N/A	N/A	N/A
1757-FFLDx linking device	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes	N/A
1757-FFLDCx linking device ⁽²⁾	N/A	N/A	Yes	Yes	No	Yes	N/A	N/A	N/A
1788HP-EN2PA-R	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes	N/A
1788HP-CN2PA-R	N/A	N/A	Yes	Yes	No	Yes	N/A	N/A	N/A
1715 Redundant I/O	No	No	No	No	No	No	Yes	Yes	N/A
1756-ENx - no connection	Yes	Yes	N/A	N/A	N/A	N/A	Yes	Yes	N/A
1756-ENx - rack-optimized	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes	N/A
Generic EtherNet/IP third party - direct	N/A	N/A	N/A	N/A	N/A	N/A	Yes	Yes	N/A
1794 FLEX™ I/O	N/A	N/A	Yes	Yes	No	No	Yes	No	Yes - Analog output modules only
POINT I/O™	N/A	N/A	Yes	Yes	No	No	Yes	No	Yes ⁽³⁾

(1) Support for I/O modules added with RSLogix 5000 software, version 15.00.00.

(2) Support for 1757-FFLDC devices added with RSLogix 5000 software, version 17.00.00.

(3) When you lose communication to the controller, POINT I/O modules ignore the last output state configuration, and set the outputs to zero.

Design Considerations for Runtime/Online Addition of Modules

When you design your network, address these considerations to add modules at runtime.

Design Issue	Considerations
I/O modules	When planning to add 1756 I/O modules at runtime, leave space in the local chassis, remote chassis on a ControlNet network, or remote chassis on an EtherNet/IP network for the I/O modules you want to add.
Other modules	You can add 1757-FFLDC devices remotely via the unscheduled portion of a ControlNet network at runtime.
Input transmission rate	Make sure the RPIs work for the data you want to send and receive. Make sure the added I/O does not depend on change of state data.
Network topology	On a ControlNet network, install spare taps so you can add modules at runtime without disrupting the network. Each tap must be terminated so as to not ground out the system. Check ControlNet system requirements to determine how many spare taps your network can support. <ul style="list-style-type: none"> • In a ControlNet network with redundant cabling, you can break the trunk and add a tap, but redundant cabling is lost during the module installation. • In a ControlNet ring, add a drop off the ring or add new nodes off the coax and disrupt only part of the network. • You could remove an existing node and add a repeater off that drop. Then reconnect the existing node and add any new nodes off the new segment. On an EtherNet/IP network, reserve some connection points on the switch so that you can connect additional nodes or switches in the future.
Network configuration	On a ControlNet network, plan which communication can be scheduled or unscheduled. On an EtherNet/IP network, all communication is immediate and occurs based on a module's RPI (also referred to as unscheduled). If you know that you need a new chassis with digital modules in the future, configure the network and add it to the I/O configuration tree as rack optimized. Then inhibit the communication adapter until you need the chassis.
Network performance	You can add modules at runtime until you impact the capacity of the communication module. Make sure that you have sufficient communication modules for the connections you plan to add.

For more information, see the Runtime/Online Addition of ControlLogix (1756) I/O Over ControlNet and EtherNet/IP White Paper, publication [LOGIX-WP006](#).

Notes:

Determine the Appropriate Network

EtherNet/IP™, ControlNet™, and DeviceNet™ networks share a universal set of communication services. These are the recommended networks for Logix control systems.

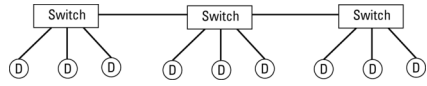
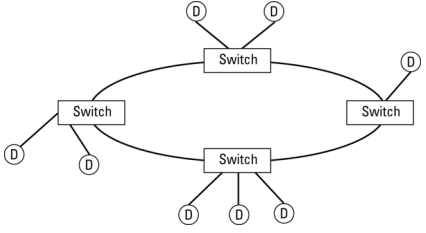
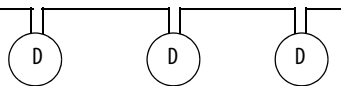
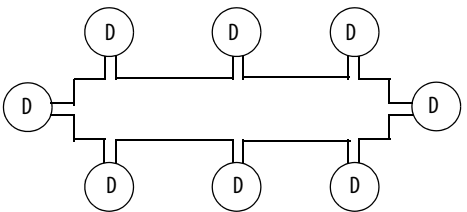
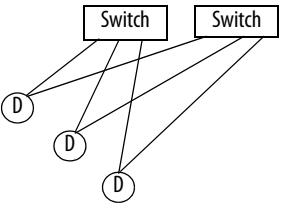
Comparison	EtherNet/IP Network	ControlNet Network	DeviceNet Network
Function	Plant management system tie-in (material handling); configuration, data collection, and control on a high-speed network	Supports transmission of time critical data between PLC processors and I/O devices	Connects low-level devices directly to plant-floor controllers—without interfacing them through I/O modules
Typical devices networked 1756-RM094	Mainframe computers Programmable controllers Robots HMI I/O Drives Process instruments	Programmable controllers I/O chassis HMIs PCs Drives Robots	Sensors Motor starters Drives PCs Push buttons Low-end HMIs Bar code readers PLC processors Valve manifolds
Data repetition	Large packets, data sent regularly	Medium-size packets; data transmissions are deterministic and repeatable	Small packets; data is sent as needed
Number of nodes, max	No limit	99 nodes	64 total nodes
Data transfer rate	10 Mbps, 100 Mbps, or 1 Gbps	5 Mbps	500 Kbps, 250 Kbps, or 125 Kbps
Typical use	Plant-wide architecture High-speed applications	Redundant applications Scheduled communication	Supply power and connectivity to low-level devices.

Follow these guidelines when planning a network.

Design Issue	Considerations
Network topology	Plan for future connections. Plan for additional controllers and/or communication modules to handle future I/O modules. Consider distances between devices. Determine resiliency requirements.
Network configuration	On a ControlNet network, plan which communication can be scheduled or can be unscheduled. On an EtherNet/IP network, all I/O communication is based on a the RPI of the module.
Network performance	Make sure that you have sufficient communication modules for the connections you plan to use. Use available network performance tools.
Chassis	Consolidate communication connections for multiple modules to one network node. Group digital I/O modules into a rack-optimized connection to reduce the amount of communication and network bandwidth.
Input transmission rate	Make sure the RPIs work for the data you want to send and receive. Make sure that I/O added at runtime does not depend on change of state data.

For more information about planning for adding I/O modules at runtime/online, see [Runtime/Online Addition of Modules on page 83](#).

EtherNet/IP Network Topology

EtherNet/IP Network	Topology
<ul style="list-style-type: none"> • An EtherNet/IP network supports messaging, produced and consumed tags, and distributed I/O • An EtherNet/IP network supports half-duplex/full-duplex, 10 Mbps or 100 Mbps operation • An EtherNet/IP network requires no network scheduling • There are several methods available to configure EtherNet/IP network parameters for devices. Not all methods are always available. These methods are device and configuration dependent: <ul style="list-style-type: none"> – DHCP – Rockwell Automation BOOTP/DHCP utility – RSLinx software – Logix Designer application – RSNetWorx™ for EtherNet/IP software – Web browser – SNMP tools <p>Application Ideas</p> <ul style="list-style-type: none"> • Connectivity to commercial devices (such as cameras and phones) • Business systems with remote access or sharing data • Applications with motion or safety on the same network. • Plant management (material handling) • Configuration, data collection, and control on a high-speed network • Time-critical applications with no established schedule • Inclusion of commercial technologies (such as video over IP) • Internet/Intranet connection 	<p>Star</p>  <p>Ring with Switches</p>  <p>Linear</p>  <p>Device Level Ring</p>  <p>Redundant Star</p> 

Guidelines for EtherNet/IP Networks

Guideline	Description						
Use these publications.	<ul style="list-style-type: none"> EtherNet/IP Network Configuration User Manual, publication ENET-UM001 EtherNet/IP Embedded Switch Technology Application Guide, publication ENET-AP005 EtherNet/IP Design Considerations Reference Manual, publication ENET-RM002 						
Make sure that the switch has the required features.	<p>For EtherNet/IP control, use an industrial-grade switch.</p> <table border="1"> <thead> <tr> <th>Required or Recommended</th> <th>Switch Feature</th> </tr> </thead> <tbody> <tr> <td>Required</td> <td>Full-duplex capability on all ports</td> </tr> <tr> <td>Recommended</td> <td> <ul style="list-style-type: none"> VLAN Autonegotiation and manually configurable speed/duplex Wire-speed switching fabric SNMP IGMP snooping constrains multicast traffic to ports associated with a specific IP multicast group Port diagnostics Port mirroring (required for troubleshooting) STP for loop prevention QoS Network Address Translation (NAT) </td> </tr> </tbody> </table>	Required or Recommended	Switch Feature	Required	Full-duplex capability on all ports	Recommended	<ul style="list-style-type: none"> VLAN Autonegotiation and manually configurable speed/duplex Wire-speed switching fabric SNMP IGMP snooping constrains multicast traffic to ports associated with a specific IP multicast group Port diagnostics Port mirroring (required for troubleshooting) STP for loop prevention QoS Network Address Translation (NAT)
Required or Recommended	Switch Feature						
Required	Full-duplex capability on all ports						
Recommended	<ul style="list-style-type: none"> VLAN Autonegotiation and manually configurable speed/duplex Wire-speed switching fabric SNMP IGMP snooping constrains multicast traffic to ports associated with a specific IP multicast group Port diagnostics Port mirroring (required for troubleshooting) STP for loop prevention QoS Network Address Translation (NAT) 						
Data transmission depends on the controller.	<p>The type of Logix5000 controller determines the data transmission rate.</p> <ul style="list-style-type: none"> ControlLogix and SoftLogix controllers transmit data at the RPI you configure for the module. CompactLogix controllers transmit data at powers of 2 ms (such as 2, 4, 8, 16, 64, or 128). For example, if you specify an RPI of 100 ms, the data actually transfers at 64 ms. 						
You can add I/O modules at runtime.	<p>With Logix controller firmware, revision 15, you can add 1756 I/O modules to remote chassis connected via an EtherNet/IP network to a running controller. You can configure direct or rack-optimized connections. For more information see Runtime/Online Addition of Modules on page 83.</p>						
Data transmission rate depends on the RPI.	<p>An EtherNet/IP network broadcasts I/O information to the controller based on the RPI setting. With change of state (COS) enabled and:</p> <ul style="list-style-type: none"> No data changes, the EtherNet/IP module produces data every RPI. Data changes, the EtherNet/IP module produces data at a maximum rate of RPI/4. 						
Select unicast EtherNet/IP communication whenever possible.	<p>To reduce bandwidth use and preserve network integrity, some facilities block multicast Ethernet packets. Multicast is a more efficient method for transmitting data with multiple consumers and redundancy applications.</p> <p>You can configure multicast or unicast connections for:</p> <ul style="list-style-type: none"> Produced and consumed tags by using the Logix Designer application I/O modules by using the Logix Designer application. <p>Unicast connections help with the following:</p> <ul style="list-style-type: none"> Letting produced and consumed tag communication span multiple subnets Reduce network bandwidth. Simplify configuration for EtherNet/IP network devices because of unicast default setting for the Logix Designer application. 						
Network Address Translation (NAT)	<p>Network Address Translation is a service that can translate a packet from one IP address to another IP address. NAT translates one IP address to another IP address via a NAT-configured switch. The switch translates the source and destination addresses within data packets as traffic passes between subnets.</p> <p>This service is useful if you reuse IP addresses throughout a network. NAT enables devices that share one IP address on a private subnet to be segmented into multiple identical private subnets while maintaining unique identities on the public subnet. The terms private and public differentiate the two networks on either side of the NAT device. The terms do not mean that the public network must be Internet routable.</p> <p>The implementation of NAT in Stratix™ switches is distinct in these ways:</p> <ul style="list-style-type: none"> One-to-one NAT—The switch uses one-to-one NAT, rather than one-to-many NAT. One-to-one NAT requires that each source address translates to one unique destination address. Unlike one-to-many NAT, multiple source addresses cannot share a destination address. Layer 2 implementation—The implementation of NAT operates at the Layer 2 (MAC) level. At this level, the switch can replace only IP addresses and does not act as a router. 						

Guidelines for Switches in EtherNet/IP Systems

Use a Managed Switch	Use an Unmanaged Switch
<ul style="list-style-type: none"> The EtherNet/IP control system is directly connected to the business system via a switch or router. Proper segregation of the control and business network is always a good design practice. The system has non-Rockwell Automation EtherNet/IP devices that are connected on the network (except for personal computers). These devices can fail to properly handle the multicast traffic that is generated by the I/O devices. If the system performs troubleshooting, you need port mirroring, which is only supported with a managed switch. 	<p>In I/O Systems—Only in These Cases</p> <ul style="list-style-type: none"> In an isolated EtherNet/IP architecture, the control system is not directly connected to the business system. Or, the control system is connected to the business system via a ControlLogix gateway (for example, a ControlLogix chassis contains two 1756-ENBT modules; one is connected to the control system and the other is connected to the business system). The EtherNet/IP control system contains only Rockwell Automation devices (except for personal computers). Traffic loading through each device (in packets/sec) is less than the capacity of each device. <p>In Non-I/O Systems</p> <ul style="list-style-type: none"> The EtherNet/IP traffic on the network consists of messaging only (MSG instructions, HMI, program upload/download).

If you use an unmanaged switch, you give up these features:

- Switch port diagnostics
- Port mirroring
- Forced duplex speed
- SNMP
- IGMP snooping
- Web browser to view configuration and diagnostics
- STP or loop prevention
- QoS for network prioritization

Determine Whether Your System Operates Properly

Rockwell Automation EtherNet/IP devices have embedded diagnostic web pages.

On this web page	Look for
Ethernet Statistics page <ul style="list-style-type: none"> All media counters In Error and Out Error counters Rejected Packets counter 	Numbers near zero (0) and not incrementing.
Diagnostic Overview page	<ul style="list-style-type: none"> An absence of connection timeouts. Packets/sec counts that are within the capacity of each device. MISSED counter-values that are under I/O Packet Counter Statistics that are zero.

If connections frequently break or if HMIs appear to update slowly, reduce traffic loading. If the situation is multicast-related, it can also help to use managed switches with IGMP snooping.

Stratix Industrial Switches

Switch	Description
Stratix 8000™ and 8300 - modular managed with Cisco technology	<ul style="list-style-type: none"> Optimized for the enterprise and plant floor Stratix 8000 for layer 2; Stratix 8300™ for layer 3 routing capability IT friendly - Cisco operating system, feature set, and user interface Engineer friendly – Logix-based configuration, Logix-based tags, and FactoryTalk® View faceplates
Stratix 5700™ - managed	<ul style="list-style-type: none"> Compact size Layer 2 access switch IT friendly - Cisco operating system, feature set, and user interface Engineer friendly – Logix-based configuration, Logix-based tags, and FactoryTalk® View faceplates
Stratix 6000™ - fixed managed	<ul style="list-style-type: none"> Optimized for the plant floor Engineer friendly – Logix-based configuration, Logix-based tags, and FactoryTalk View faceplates
Embedded technology	<ul style="list-style-type: none"> Two Ethernet ports that are embedded in Allen-Bradley products Enables high-speed ring and linear topologies No configuration required 1783-ETAP, 1783-ETAP1F, 1783-ETAP2F – Ethernet tap modules for connectivity to single-port devices
Stratix 2000™ - fixed unmanaged	<ul style="list-style-type: none"> Compact size IP20 versions No configuration required

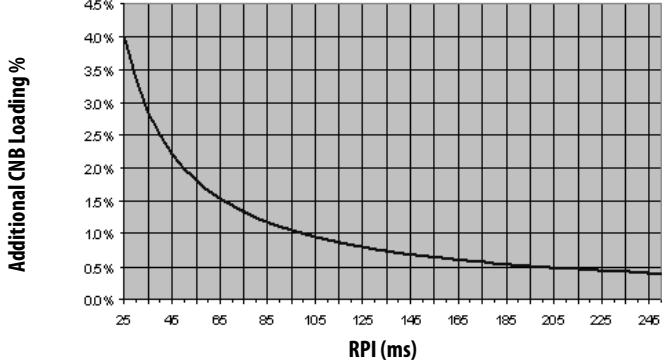
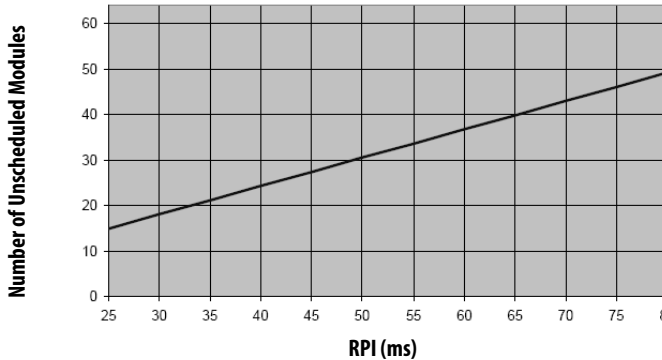
ControlNet Network Topology

ControlNet Network	Topology
<ul style="list-style-type: none"> A ControlNet network lets both I/O and messaging on the same wire. Multiple controllers and their respective I/O can also be placed on the same ControlNet wire. When new I/O is added, or when the communication structure on an existing I/O module changes, you must use RSNetWorx for ControlNet software to reschedule the network. If the network timing changes, every device with scheduled traffic on the network is affected. To reduce the impact of changes, place each CPU and its respective I/O on isolated ControlNet networks. Place shared I/O and produced/consumed tags on a common network available to each CPU that needs the information. Built-in redundant cabling supports I/O network and provides HMI switchover in redundant ControlLogix system. <p>Application Ideas</p> <ul style="list-style-type: none"> Default Logix network Best replacement for Universal Remote I/O Backbone to multiple distributed DeviceNet networks Peer interlocking network Common devices include: Logix5000 controllers, PanelView terminals, I/O modules, and drives 	

Guidelines for ControlNet Networks

Guideline	Description
Use these publications.	<ul style="list-style-type: none"> ControlNet Coax Media Planning and Installation Guide, publication CNET-IN002 ControlNet Fiber Media Planning and Installation Guide, publication CNET-IN001 ControlNet Network Configuration User Manual, publication CNET-UM001
Adjust the default RSNetWorx for ControlNet settings.	<p>Change these settings in the RSNetWorx for ControlNet software:</p> <ul style="list-style-type: none"> UMAX (highest unscheduled node on the network) <ul style="list-style-type: none"> Default is 99 The network takes the time to process the total number of nodes that are specified in this setting, even if there are not that many devices on the network Change to a reasonable level to accommodate the active network devices and additional devices that can be connected SMAX (highest scheduled node on the network) <ul style="list-style-type: none"> Default is 1 This must be changed for all systems Set SMAX < UMAX
Design for at least 400 KB of available, unscheduled network bandwidth, as displayed by RSNetWorx for ControlNet software.	<p>Leaving too little bandwidth for unscheduled network communication results in poor message throughput and slower workstation response.</p> <p>Unscheduled data transfers on ControlNet occur asynchronous to the program scan and support a maximum of 510 bytes/node per ControlNet NUT.</p>
Place DeviceNet (1756-DNB) communication modules in the local chassis.	<p>DeviceNet (1756-DNB) communication modules have multiple, 500-byte data packets that impact scheduled bandwidth. Placing these modules in the same chassis as the controller avoids this data being scheduled over the DeviceNet network. If you must place these communication devices in remote chassis, configure the input and output sizes to match the data that is configured in RSNetWorx for DeviceNet software. This reduces the amount of data that must be transmitted.</p>
Limit 1756-CNB, 1756-CNBR connections.	<p>For best performance, limit the 1756-CNB, 1756-CNBR to 40...48 connections. Add additional 1756-CNB, 1756-CNBR modules in the same chassis if you need more connections. To improve system performance, you can add more modules and split connections among the modules.</p> <p>If the chassis that contains the CNB module also contains multiple digital I/O modules, configure the CNB communication format for Rack Optimization. Otherwise, use None.</p> <p>As a cost savings measure, use 1756-CNB, 1756-CNBR modules in chassis that contain only I/O modules for traditional adapter functionality. Use the 1756-CN2, 1756-CN2R, 1756-CN2RXT modules in the same chassis as the controller for traditional scanner functionality.</p>
For additional connections, consider the 1756-CN2, 1756-CN2R, 1756-CN2RXT modules.	<p>The 1756-CN2/B, 1756-CN2R/B, 1756-CN2RXT communication modules each support 131 connections, and have higher performance than previous modules.</p> <p>The 1756-CN2/A, 1756-CN2R/A communication modules each support 100 connections.</p>
If you change network settings, resave each controller project.	<p>Any time that you edit the network with RSNetWorx for ControlNet software and you save or merge your edits, connect to each controller in the system with their respective project file and perform a save and upload. This copies the ControlNet settings into the offline, database file and makes sure that future downloads of the controller permit it to go online without having to run RSNetWorx for ControlNet software.</p>
You can add I/O modules at runtime.	<p>With Logix controller firmware, revision 15, you can add 1756 I/O modules and some drives to remote chassis connected via ControlNet to a running controller. It is recommended to use a 1756-CN2/B, 1756-CN2R/B, or 1756-CN2RXT module as the traditional scanner in these applications.</p>
Data transmission depends on the controller.	<p>The type of Logix5000 controller determines the data transmission rate.</p> <ul style="list-style-type: none"> ControlLogix and SoftLogix controllers transmit data at the RPI you configure for the module. CompactLogix controllers transmit data at powers of 2 ms (such as 2, 4, 8, 16, 64, and 128). For example, if you specify an RPI of 100 ms, the data actually transfers at 64 ms.

Guidelines for Unscheduled ControlNet Networks

Guideline	Description																										
<p>You can run an entire ControlNet network as unscheduled.</p>	<p>An unscheduled ControlNet network:</p> <ul style="list-style-type: none"> • provides for easier network configuration. • is useful if your I/O updates needs are slower. • supports the addition of 1756 I/O modules and some drives without placing the controller in Program mode. • provides an HMI network with fast switchover times in a redundant controller system. <p>You still must run RSNetWorx for ControlNet software at least once to configure NUT, SMAX, UMAX, and media configuration settings.</p>																										
<p>Plan appropriately if you place I/O on an unscheduled ControlNet network.</p>	<p>Follows these recommendations for I/O on an unscheduled ControlNet network:</p> <ul style="list-style-type: none"> • A 1756-CN2, 1756-CN2R series B or later, or a 1756-CN2RXT is recommended. • Disable the Change of State (COS) feature on digital input modules because it can cause inputs to be sent faster than the RPI. • Set the real-time sample (RTS) on analog cards slower than the RPI • Dedicate a ControlNet network to I/O only. • Do not exceed 80% utilization of a 1756-CN2, 1756-CN2R, 1756-CN2RXT communication module. • Do not exceed 75% utilization of a 1756-CNB, 1756-CNBR communication module. • Have no more than 48 connections on the 1756-CNB, 1756-CNBR communication module. • Use a NUT of 10 ms or more. • Keep the SMAX and UMAX values as small as possible. 																										
<p>1756-CNB, 1756-CNBR only Set the RPI at 25 ms or slower.</p>	<p>Use RPI of 25 ms or slower for unscheduled modules to avoid overload on the 1756-CNB, 1756-CNBR communication module. Depending on the RPI, the communication module loading increases 1...4% for each I/O module added.</p> <p style="text-align: center;">Additional 1756-CNB, 1756-CNBR Loading</p>  <table border="1"> <caption>Data for Additional 1756-CNB, 1756-CNBR Loading</caption> <thead> <tr> <th>RPI (ms)</th> <th>Additional CNB Loading %</th> </tr> </thead> <tbody> <tr><td>25</td><td>4.5</td></tr> <tr><td>45</td><td>3.0</td></tr> <tr><td>65</td><td>2.0</td></tr> <tr><td>85</td><td>1.5</td></tr> <tr><td>105</td><td>1.2</td></tr> <tr><td>125</td><td>1.0</td></tr> <tr><td>145</td><td>0.8</td></tr> <tr><td>165</td><td>0.7</td></tr> <tr><td>185</td><td>0.6</td></tr> <tr><td>205</td><td>0.55</td></tr> <tr><td>225</td><td>0.5</td></tr> <tr><td>245</td><td>0.5</td></tr> </tbody> </table>	RPI (ms)	Additional CNB Loading %	25	4.5	45	3.0	65	2.0	85	1.5	105	1.2	125	1.0	145	0.8	165	0.7	185	0.6	205	0.55	225	0.5	245	0.5
RPI (ms)	Additional CNB Loading %																										
25	4.5																										
45	3.0																										
65	2.0																										
85	1.5																										
105	1.2																										
125	1.0																										
145	0.8																										
165	0.7																										
185	0.6																										
205	0.55																										
225	0.5																										
245	0.5																										
<p>1756-CNB, 1756-CNBR only The RPI affects how many I/O modules you can have.</p>	<p>This chart shows the number of modules and associated RPIs so that you do not exceed 75% utilization of the 1756-CNB, 1756-CNBR communication module.</p> <p style="text-align: center;">Maximum Number of I/O Modules in an Unscheduled Network</p>  <table border="1"> <caption>Data for Maximum Number of I/O Modules in an Unscheduled Network</caption> <thead> <tr> <th>RPI (ms)</th> <th>Number of Unscheduled Modules</th> </tr> </thead> <tbody> <tr><td>25</td><td>15</td></tr> <tr><td>30</td><td>18</td></tr> <tr><td>35</td><td>22</td></tr> <tr><td>40</td><td>26</td></tr> <tr><td>45</td><td>30</td></tr> <tr><td>50</td><td>34</td></tr> <tr><td>55</td><td>38</td></tr> <tr><td>60</td><td>42</td></tr> <tr><td>65</td><td>46</td></tr> <tr><td>70</td><td>50</td></tr> <tr><td>75</td><td>54</td></tr> <tr><td>80</td><td>58</td></tr> </tbody> </table>	RPI (ms)	Number of Unscheduled Modules	25	15	30	18	35	22	40	26	45	30	50	34	55	38	60	42	65	46	70	50	75	54	80	58
RPI (ms)	Number of Unscheduled Modules																										
25	15																										
30	18																										
35	22																										
40	26																										
45	30																										
50	34																										
55	38																										
60	42																										
65	46																										
70	50																										
75	54																										
80	58																										

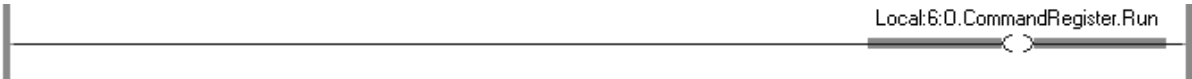
Compare Scheduled and Unscheduled ControlNet Communication

Scheduled ControlNet Communication	Unscheduled ControlNet Communication
Deterministic	Less deterministic than scheduled communication Provides simpler ControlNet installations when scheduled networks are not required
To add scheduled I/O on the ControlNet network, you must: <ul style="list-style-type: none"> • Add the I/O to an offline controller project. • Download the project to the controller. • Run RSNetWorx to schedule the network requires network to be scheduled (must stop the network and put the controller in Program mode to schedule a network). • Save the controller project. 	Can be changed online without impacting the schedule New modules can affect other modules communicating via unscheduled bandwidth Supports 1756 I/O modules and some drives
RPI and NUT determine module communication rates	RPI determines module communication rates
MSG and HMI traffic can occur on the same network because they are isolated in unscheduled traffic MSG and HMI traffic do not affect I/O communication	Recommend 1756-CN2, 1756-CN2R, 1756-CN2RXT Recommend a dedicate ControlNet network for only I/O modules MSG and HMI traffic can affect I/O communication
Direct and rack-optimized connections to I/O	Only direct connections to I/O (results in being able to use fewer total I/O modules because of the connection limits of controllers and communication modules)
Supports any firmware revision of a ControlNet communication module	You can use any 1756-CN2, 1756-CN2R, 1756-CN2RXT communication module If you use a 1756-CNB, 1756-CNBR communication module, it must be series D or later
Supports any I/O platform that can communicate via a ControlNet network	Supports only 1756 I/O modules

DeviceNet Network Topology

DeviceNet Network	Topology
<ul style="list-style-type: none"> • You need a DeviceNet scanner to connect the controller to DeviceNet devices. • You must use RSNetWorx for DeviceNet software to configure devices and create the scanlist for the scanner. • You can configure the network communication rate as 125 Kbps (default and a good starting point), 250 Kbps, or 500 Kbps. • If each device on the network (except the scanner) sends ≤ 4 bytes of input data and receives ≤ 4 bytes of output data, you can use AutoScan to configure the network. <p>Application Ideas</p> <ul style="list-style-type: none"> • Distributed devices • Drives network • Diagnostic information 	<p>Single Network</p> <pre> graph TD CPU[CPU] --- Bus1[] Scanner[Scanner] --- Bus1 Bus1 --- Device1[Device] Bus1 --- Device2[Device] Bus1 --- Device3[Device] Bus1 --- Device4[Device] Bus1 --- Device5[Device] </pre> <p>Several Smaller Distributed Networks (Subnets)</p> <pre> graph TD CPU[CPU] --- Bus2[] LD1[Linking Device] --- Bus2 LD2[Linking Device] --- Bus2 Bus2 --- Device1_1[Device] Bus2 --- Device1_2[Device] Bus2 --- Device1_3[Device] Bus2 --- Device2_1[Device] Bus2 --- Device2_2[Device] Bus2 --- Device2_3[Device] </pre>

Guidelines for DeviceNet Networks

Guideline	Description
Use these publications.	<ul style="list-style-type: none"> • DeviceNet Cable System Manual, publication DNET-UM072 • DeviceNet Network Configuration User Manual, publication DNET-UM004
Use the DeviceNet Tag Generator tool.	<p>The Logix Designer application includes a DeviceNet tag generator tool that creates device-specific structured tags and logic based on the network configuration in RSNetWorx for DeviceNet software.</p> <p>The logic copies data to and from the DNB data array tags to the device tags so that data is presented synchronously to program scan.</p>
Place DeviceNet (DNB) communication modules in the local chassis.	<p>Placing DNB modules in the local chassis maximizes performance, especially in ControlLogix systems.</p> <p>Size the input and output image for the DNB modules to the actual devices that are connected plus 20% for future growth. If you have to place DNB modules in remote chassis, sizing the input and output images is critical for best performance.</p>
Verify the total network data does not exceed the maximum DNB data table size.	<p>A DNB supports:</p> <ul style="list-style-type: none"> • 124, 32-bit input words. • 123, 32-bit output words. • 32, 32-bit status words. <p>You can use RSNetWorx for DeviceNet software offline to estimate network data. Use a second DNB if there is more network data than one module can support.</p>
Set up slaves first.	<p>Configure a device's parameters before adding that device to the scanlist. You cannot change the configuration of many devices once they are already in the scanlist.</p> <p>If you configure the scanner first, there is a chance that the scanner configuration cannot match the current configuration for a device. If the configuration does not match, the device does not show up when you browse the network.</p>
Leave node address 63 open to add nodes.	<p>Devices default to node 63 out-of-the-box. Leave node address 63 unused so you can add a new devices to the network. Then change the address of the new device.</p>
Leave node address 62 open to connect a computer.	<p>Always leave at least one open node number to let a computer be attached to the network if needed for troubleshooting or configuration.</p>
Don't forget to set the scanner run bit.	<p>For the scanner to be in Run mode, the controller must be in Run mode and the logic in the controller must set the scanner's run bit.</p>
	
Make sure you have the most current EDS files for your devices.	<p>RSNetWorx for DeviceNet software uses EDS file to recognize devices. If the software is not properly recognizing a device, you are missing the correct EDS files) For some devices, you can create an EDS file by uploading information from the device. Or you can get EDS files from: http://www.ab.com/networks/eds.</p>

Notes:

Communicate with Other Devices

The MSG instruction asynchronously reads or writes a block of data to another device.

If the target device is a	Select one of these message types
Logix5000 controller	CIP Data Table Read
	CIP Data Table Write
I/O module that you configure with the Logix Designer application	Module Reconfigure
	CIP Generic
SERCOS drive	SERCOS IDN Read
	SERCOS IDN Write
PLC-5 controller	PLC5 Typed Read
	PLC5 Typed Write
	PLC5 Word Range Read
	PLC5 Word Range Write
SLC controller MicroLogix™ controller	SLC Typed Read
	SLC Typed Write
Block transfer module	Block Transfer Read
	Block Transfer Write
PLC-3® processor	PLC3 typed read
	PLC3 typed write
	PLC3 word range read
	PLC3 word range write
PLC-2® processor	PLC2 unprotected read
	PLC2 unprotected write

Cache Messages

Some types of messages use a connection to send or receive data. Some also give you the option to either leave the connection open (cache) or close the connection when the message is done transmitting. This table shows messages that use a connection and whether you can cache the connection.

Message Type	Communication Method	Uses Connection	Can Cache Connection
CIP data table read or write	CIP	Yes	Yes
PLC2, PLC3, PLC5, or SLC (all types)	CIP		
	CIP with Source ID		
	DH+	Yes	Yes
CIP generic	N/A	Your option ⁽¹⁾	Your option ⁽¹⁾
Block transfer read or write	N/A	Yes	Yes

(1) You can connect CIP generic messages, but for most applications we recommend that you leave CIP generic messages unconnected.

A cached connection remains open until one of the following occurs:

- The controller goes to Program mode.
- You rerun the message as uncached.
- Another message is initiated and a cached buffer is needed.
- An intermediate node in the connection goes down.

Message Buffers

A Logix5000 controller has buffers for unconnected messages and for cached messages. Buffers store incoming and outgoing message data until the controller can process the data.

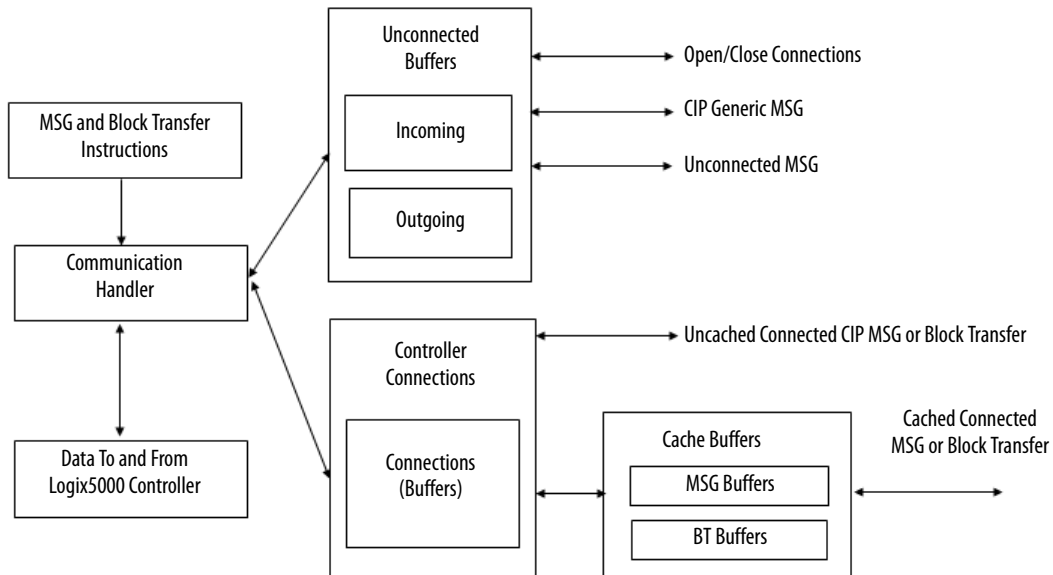


Table 5 - Message Buffer Guidelines

Buffer	Description
<p>Ten outgoing unconnected buffers (20 on 1756-L7x controllers)</p> <p>You can increase this to 40 by using a CIP Generic message instruction. Each buffer that you add uses approximately 1.2 KB of I/O memory. See the MSG section in the Logix 5000 Controllers General Instructions Reference Manual, publication 1756-RM003.</p>	<p>The outgoing unconnected buffers are for:</p> <ul style="list-style-type: none"> Establishing I/O connections to local I/O modules and remote devices on ControlNet, EtherNet/IP, and remote I/O networks. Executing unconnected PLC2, PLC3, PLC5, or SLC (all types) messages over Ethernet/IP or ControlNet (CIP and CIP with Source ID) networks. Initiation of messaging over a DH+ network (uses 2 buffers, one to open the connection and one to transfer data). Initiation of uncached block transfers. Initiation of uncached CIP read/write message instructions. Initiation of cached block transfers. Initiation of cached CIP read/write messages instructions. CIP Generic message instructions.
<p>Three incoming unconnected buffers</p>	<p>The incoming unconnected buffers are for:</p> <ul style="list-style-type: none"> Initial receiving of a cached CIP message instruction. Receiving an uncached CIP message instruction. Receiving a message over a DH+ network. Receiving a CIP Generic message instruction. Receiving a read or write request from a ControlNet PanelView™ terminal (unconnected messaging). Initial receiving of a read request from an EtherNet/IP PanelView terminal (connected messaging). Receiving a write request from an EtherNet/IP PanelView terminal (unconnected messaging). Receiving an initial request from the Logix Designer application to go online. Initial receiving of RSLinx connections.
<p>Cached buffers</p> <p>Revision 12 and later firmware: 32 cached buffers for any combination of messages and block transfers</p> <p>Revision 11 and earlier firmware: 16 cached buffers only for messages and 16 cached buffers only for block transfers</p>	<p>The cached buffers are outgoing buffers for messages and block transfers. A cached connection helps message performance because the connection is left open and does not need to be re-established the next time that it is executed. A cached connection counts towards the total limit of connections for a controller. A cached connection is refreshed at the connection RPI. All cached entries are closed when the controller transitions to Program mode. With revision 12 and later firmware, you can cache 32 messages and block transfers (any combination). Previous revisions of controller firmware let you cache 16 messages only and 16 block transfers only.</p> <p>The first time a cached message is executed, it uses one of the 10 outgoing unconnected buffers. When the connection is established, it moves into the cached buffer area.</p> <p>For optimum performance, do not cache more messages or block transfers than there are cached buffers. If you cache more than the available cached buffers, the controller looks for a connection that has been inactive for the longest time, closes that connection, and lets a new connection take its place. The controller closes a cached message or block transfer, depending on which has been inactive the longest. If all 32 cached connections are in use, the message is executed as connected and, once it is completed, the connection is closed.</p> <p>You can multiplex cached connections. If a connection is inactive and a message instruction executes that has the same target and path, it uses that inactive connection. For example, if you have a block transfer read and write to the same module, interlock the read and write so that only one is active at a time. Then when they are cached, they use the same cached connection.</p>

Outgoing Unconnected Buffers

Buffers	Use
1...10	The first 10 buffers (default) are shared for unconnected messaging, initiating connected messaging, establishing I/O connections, and establishing produced/consumed connections.
11	The 11th buffer is dedicated to establishing I/O and produced/consumed connections.
12...40	The 12th to the 40th buffers are used only for initiating connected messages and executing unconnected messages. To increase the outgoing buffers to a value higher than 11, execute a CIP generic message to configure that change each time you transition from Program mode to Run mode.

Guidelines for Messages

Guideline	Description
Message tags must exist as controller-scoped, base tags.	The operating system accesses the information in a message tag asynchronously to the program scan. Along with the visible fields within the message tag, there are hidden attributes that are only referenced by the background operating system.
You can have more than 32 messages in a program.	The controller supports 32 active, cached messages at a time. If you determine that there are more than 32 messages, you cannot cache all messages. You need extra programming to help ensure that no more than 32 messages are active simultaneously. Before controller revision 12, the controller supported 16 active, cached messages at a time.
You can use a message to send a large amount of data.	Even though there are network packet limitations (such as 500 bytes on ControlNet and 244 bytes on DH+), the controller can send a large amount of data from one MSG instruction. When configuring the message, select an array as the source/destination tags and select the number of elements (as many as 32,767 elements) you want to send. The controller automatically breaks the array into small fragments and sends all of the fragments to the destination. On the receiving side, the data appears in fragments, so some application code can be required to detect the arrival of the last piece.
Do not manipulate the message status bit	Do not change the following status bits of a MSG instruction: <ul style="list-style-type: none"> • DN • EN • ER • EW • ST Do not change those bits either by themselves or as part of the FLAGS word. If you do, the controller can have a nonrecoverable fault. The controller clears the project from its memory when it has a nonrecoverable fault.

Guidelines to Manage Message Connections

Guideline	Description
Create user-defined structures or arrays.	User-defined structures let you organize your data to match your machine or process. <ul style="list-style-type: none"> • One tag contains all data that is related to a specific aspect of your system. This keeps related data together and easy to locate, regardless of its data type. • Each individual piece of data (member) gets a descriptive name. This automatically creates an initial level of documentation for your logic. • You can use the structure to create multiple tags with the same data layout. • RSLinx can optimize user-defined structures more than standalone tags.
Cache connections when appropriate.	If a message executes repeatedly, cache the connection. This keeps the connection open and optimizes execution time. Opening a connection each time the message executes increases execution time. If a message executes infrequently, do not cache the connection. This closes the connection upon completion of the message, which frees up that connection for other uses.

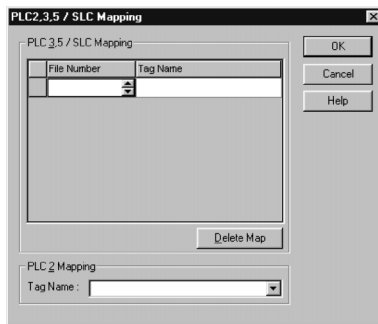
The system overhead timeslice percentage you configure for the controller determines the percentage of controller time (excluding the time for periodic and event tasks) that is devoted to communication and background functions. This includes sending and receiving messages.

For more information, see [Select a System Overhead Percentage on page 31](#).

Guidelines for Block Transfer Messages

Guideline	Description
Distribute 1771 analog modules across multiple chassis.	Distributing 1771 analog modules across multiple chassis reduces the number of block transfers that one 1771-ACN or 1771-ASB module manages.
Isolate different 1771 chassis on different networks.	Isolating different chassis onto different networks diversifies the communication so that no single network or communication module has to deal with all communication.
Increase ControlNet unscheduled bandwidth.	If communicating over a ControlNet network, increase the amount of ControlNet unscheduled bandwidth to permit additional time on the network for data exchange. See Compare Scheduled and Unscheduled ControlNet Communication on page 94 for more information about unscheduled bandwidth on a ControlNet network.
Increase the system overhead timeslice percentage.	Increase the system overhead timeslice to allocate more CPU time to communication processing from the continuous task.
Interlock block transfer read and write messages to the same module.	Programmatically interlock block transfer read and write messages to the same module so that both operations cannot be active simultaneously.
Use the 1757-ABRIO module for systems with a high number of block transfer modules.	The 1757-ABRIO module provides connectivity from a ControlLogix chassis to 1771 I/O and other modules that are connected via remote I/O. The 1757-ABRIO module off-loads the burden of performing block transfers from the controller and increases the number of block transfer operations that can be performed.

Map Tags



A Logix5000 controller stores tag names on the controller so that other devices can read or write data without having to know physical memory locations. Many products only understand PLC/SLC data tables, so the Logix5000 controller offers a PLC/SLC mapping function that lets you map Logix tag names to memory locations.

- You only have to map the file numbers that are used in messages; the other file numbers do not need to be mapped.
- The mapping table is loaded into the controller and is used whenever a logical address accesses data.
- You can only access controller-scoped tags (global data).

Follow these guidelines when you map tags.

- Do not use file numbers 0, 1, and 2. These files are reserved for Output, Input, and Status files in a PLC-5 processor.
- Use PLC-5 mapping only for tag arrays of data type INT, DINT, or REAL. Attempting to map elements of system structures can produce undesirable effects.
- Use these file types and identifiers.

For this Logix5000 array type	Use this PLC file identifier
INT array	N or B
DINT array	L
REAL array	F

Notes:

FactoryTalk Alarms and Events System

The FactoryTalk® Alarms and Events system integrates alarming between FactoryTalk View SE applications and Logix5000 controllers by embedding an alarming engine in Logix5000 controllers. You need the following tools:

- Logix Designer application, version 21 or later (or RSLogix 5000 software, version 16 or later)
- FactoryTalk View SE, version 5.0 or later
- Logix5000 controllers

Firmware	Revision
ControlLogix non-redundant controllers	16.20 or later
ControlLogix redundant controllers	16.60 or later
CompactLogix controllers	16.20 or later
SoftLogix controllers	16.40 or later

Two Logix-based alarm instructions are available in relay ladder, structured text, and function block diagram.

- The Digital Alarm (ALMD) instruction detects alarms that are based on Boolean (true/false) conditions.
- The Analog Alarm (ALMA) instruction detects alarms that are based on the level or rate of change of analog values.

Guidelines for Logix-based Alarm Instructions

Guideline	Description
Estimate increased controller memory use for each alarm.	The alarm instructions use new alarm data types that contain state information and time stamps for each alarm. Estimate this memory use in the controller: <ul style="list-style-type: none"> • 2 KB per FactoryTalk Alarms and Events subscriber that receives alarms from the controller • There is a maximum of 16 subscribers per controller. Most applications only require one subscriber to a controller to provide data to many FactoryTalk View SE clients. • 2.2 KB per alarm (typical), depends on use of associated tags
Alarm instructions increase total controller scan time.	The ALMD instructions and ALMA instructions affect total scan time. See Logix5000 Controllers Instruction Execution Time and Memory Use Reference Manual, publication 1756-RM087 for execution times for your controller firmware. An alarm state change is any event that changes the condition of the alarm, such as acknowledging or suppressing the alarm. Minimize the potential for many alarms changing state simultaneously (alarm bursts) by creating dependencies on related alarms. Large alarm bursts can have a significant impact on application code scan time. Important: In redundancy systems, consider scan time impact due to crossloading alarm tag data. For more information see the ControlLogix Enhanced Redundancy System User Manual, publication 1756-UM535 .
You can edit or add an alarm instruction online.	Online edits of new and existing alarms are automatically sent to the subscribers. You do not have to re-subscribe to receive the updated information. Online changes automatically propagate from the controller alarm structure to the rest of the architecture.

Guideline	Description
<p>In relay ladder, how you define the alarm values on the instruction determines whether you can access those values programmatically through the alarm structure.</p>	<p>When you create an alarm instruction, you also create an alarm data type for that alarm. For example, MyDigitalAlarm of data type DigitalAlarm. In relay ladder, the following values are shown on the instruction:</p> <ul style="list-style-type: none"> • ProgAck • ProgReset • ProgDisable • ProgEnable <p>If you enter a value or assign a tag to these faceplate parameters (such as AckSection1All), the value or tag value is automatically written to the alarm structure each time the instruction is scanned.</p> <p>If you want to programmatically access the alarm structure, you must assign the structure tag to the faceplate. For example, to use MyAnalogAlarm.ProgAck in logic, assign the tag MyAnalogAlarm.ProgAck on the faceplate to the ProgAck parameter.</p>
<p>Test alarm behavior from within the Logix Designer application.</p>	<p>On the Status tab of the alarm dialog, monitor the alarm condition, acknowledge an alarm, disable an alarm, suppress an alarm, or reset an alarm. Use the dialog selections to see how an alarm behaves, without needing an operational HMI.</p>
<p>Reduce mistakes by making sure that alarms are noticed.</p>	<p>Shelving an alarm removes the alarm from the operator view for a period of time. It is like suppressing an alarm, except that shelving is time-limited. If an alarm is acknowledged while it is shelved, it remains acknowledged even if it becomes active again. It becomes unacknowledged when the shelve duration ends provided the alarm is still active at that moment.</p>
<p>Increase productivity by eliminating nuisance alarms.</p>	<p>Set a duration (ms) on the ALMA instruction to specify how long an alarm condition must exist before being reported. Apply the duration to individual, analog alarm levels.</p>
<p>High availability of alarm data helps reduce material losses.</p>	<p>The alarm log in the controller stores the last 10,000 alarm state transitions in a circular log. This replaces the alarm buffer in controllers with firmware earlier than revision 21.</p>

Changes in Logic

With Logix Designer version 21.00.00, the alarm instructions added and removed some parameters. The following parameters were removed:

- DeliveryER
- DeliveryDN
- DeliveryEN
- NoSubscriber
- NoConnection
- CommError
- AlarmBuffered
- Subscribers
- SubscNotified

The following parameters were added to support a shelve state:

- OperShelve
- ProgUnshelve
- OperUnshelve
- ShelveDuration
- MaxShelveDuration

Configure Logix-based Alarm Instructions

Option	Description														
Message string	<p>The message string (maximum of 255 characters, including embedded text) contains the information to display to the operator regarding the alarm. Besides entering text, you can also embed variable information. In the alarm message editor, select the variable that you want and add it anywhere in the message string.</p> <table border="1"> <thead> <tr> <th>Variable</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Alarm name</td> <td>Tag name of the alarm. /*S:0 %AlarmName*/</td> </tr> <tr> <td>Condition name</td> <td>State of the alarm (such as, true, false, high-high, or low). /*S:0 %ConditionName*/</td> </tr> <tr> <td>Input value</td> <td>True, false, or current value of the analog input value. /*N:5 %InputValue NOFILL DP:0*/</td> </tr> <tr> <td>Limit value</td> <td>Limit or condition that caused the alarm. /*N:5 %LimitValue NOFILL DP:0*/</td> </tr> <tr> <td>Severity</td> <td>The assigned importance of the alarm. /*N:5 %Severity NOFILL DP:0*/</td> </tr> <tr> <td>Values of associated tags</td> <td>Values of the selected tags that are delivered with the alarm. /*N:5 %Tag1 NOFILL DP:0*/</td> </tr> </tbody> </table> <p>This information is always sent with the alarm, viewable by the operator, and entered in the history log, regardless of whether you embed it in the message string.</p> <p>You cannot programmatically access the alarm message string from the alarm tag. To change the alarm message based on specific events, configure one of the associated tags as a string data type and embed that associated tag in the message.</p> <p>You can have multiple language versions of messages. You enter the different language via the import/export utility. For more information, see page 106.</p>	Variable	Description	Alarm name	Tag name of the alarm. /*S:0 %AlarmName*/	Condition name	State of the alarm (such as, true, false, high-high, or low). /*S:0 %ConditionName*/	Input value	True, false, or current value of the analog input value. /*N:5 %InputValue NOFILL DP:0*/	Limit value	Limit or condition that caused the alarm. /*N:5 %LimitValue NOFILL DP:0*/	Severity	The assigned importance of the alarm. /*N:5 %Severity NOFILL DP:0*/	Values of associated tags	Values of the selected tags that are delivered with the alarm. /*N:5 %Tag1 NOFILL DP:0*/
Variable	Description														
Alarm name	Tag name of the alarm. /*S:0 %AlarmName*/														
Condition name	State of the alarm (such as, true, false, high-high, or low). /*S:0 %ConditionName*/														
Input value	True, false, or current value of the analog input value. /*N:5 %InputValue NOFILL DP:0*/														
Limit value	Limit or condition that caused the alarm. /*N:5 %LimitValue NOFILL DP:0*/														
Severity	The assigned importance of the alarm. /*N:5 %Severity NOFILL DP:0*/														
Values of associated tags	Values of the selected tags that are delivered with the alarm. /*N:5 %Tag1 NOFILL DP:0*/														
Associated tags	<p>You can select as many as four additional tags from the controller project to associate with the alarm. These tags are sent with an alarm message to the alarm server. Associated tags can be BOOL, INT, SINT, DINT, REAL, or string data types. For example, a digital alarm for a pressure relief valve can also include information such as pump speed and system pressure, and tank temperature</p> <p>Optionally, embed the associated tags into the message text string.</p>														
Severity	<p>Use the configurable severity range from 1...1000 to rank the importance of an alarm. A severity of 1 is for low priority alarms; a severity of 1000 is for an emergency condition.</p> <p>By default, in the FactoryTalk alarm service, severities:</p> <ul style="list-style-type: none"> • 1...250 are low alarms. • 251...500 are medium alarms. • 501...750 are high alarms. • 751...1000 are urgent alarms. <p>You can configure how the FactoryTalk ranges are presented to the operator. The operator can also filter on alarm levels. For example, a maintenance engineer can filter to see only those alarms at severity 128.</p>														
Alarm class	<p>Use the alarm class to group related alarms. Specify the alarm class the same for each alarm you want in the same class. The alarm class is case-sensitive.</p> <p>For example, specify class Control Loop to group all alarms for PID loops.</p> <p>You can then display and filter alarms at the HMI based on the class. For example, an operator can display all tank alarms or all PID loop alarms.</p> <p>The alarm class does not replace subscription to specific alarms. The FactoryTalk View SE Alarm object graphics have configuration options to determine which controller alarms an operator sees.</p>														
View command	<p>Execute a command on the operator station when requested by an operator for a specific alarm. This lets an operator execute any standard FactoryTalk View command, such as call specific faceplates and displays, execute macros, access help files, and launch external applications. When the alarm condition occurs and is displayed to the operator, a button on the summary and banner displays lets the operator run an associated view command.</p>														
Defaults	<p>The Parameters tab of the alarm instruction properties lets you define values for instruction parameters. You can return the parameters to factory defaults and you can define your own set of instruction defaults. The instruction defaults you assign are defaults for only that instance of the instruction.</p>														

Multiple Language Versions of Alarm Messages

You can maintain alarm messages in multiple languages. Either enter the different languages in the associated language versions of the Logix Designer application or in an import/export (.CSV or .TXT) file.

You can access alarm message text from an import/export (.CSV or .TXT) file and add additional lines for translated versions of the original message string. Messages in different languages use ISO language codes in the TYPE column. Text for the operator is in the DESCRIPTION column. The SPECIFIER identifies the type of alarm.

TYPE	NAME	DESCRIPTION	SPECIFIER
ALMMSG:en-us	ALMA Logix Tag Name	HH alarm text for operator in English	HH
ALMMSG:en-us	ALMA Logix Tag Name	H alarm text for operator in English	H
ALMMSG:en-us	ALMA Logix Tag Name	L alarm text for operator in English	L
ALMMSG:en-us	ALMA Logix Tag Name	LL alarm text for operator in English	LL
ALMMSG:en-us	ALMA Logix Tag Name	ROC positive alarm text for operator	POS
ALMMSG:en-us	ALMA Logix Tag Name	ROC negative alarm text for operator	NEG
ALMMSG:de-ch	ALMA Logix Tag Name	HH Mitteilung für den Operator auf Deutsch	HH
ALMMSG:de-ch	ALMA Logix Tag Name	H Mitteilung für den Operator auf Deutsch	H

Use the import/export utility to create and translate message strings into multiple languages. The .TXT import/export format supports all languages, including Chinese, Japanese, and Korean. The .CSV import/export format does not support Chinese, Japanese, or Korean.

Importing and exporting messages always performs a merge. Deleting a message in a .CSV file does not delete the message from the .ACD file. To delete a message, import the .CSV file with the type, name, and specifier fields filled in but the description blank.

When viewing alarm messages at the HMI:

- There is no default language string. If message text does not exist for a specific language, FactoryTalk View software searches for the first language that has a message string and displays that text.
- Date and time format do not switch with the language. They follow the format of the operating system.
- Nonconfigurable dialogs, such as ACK with description dialog, do not switch languages. They use the language of the operating system.

Alarm Process

At powerup of the alarm system, the alarm uses this process to establish its initial connection to the controller.

1. The RSLinx Enterprise server initiates a subscription to the alarm log.
One subscription to the alarm log consumes 2 KB of controller memory.
2. The controller sends this alarm information to the subscriber:
 - Path and tag information
 - Alarm configuration
 - Message strings in all configured languages
3. Once the subscriber receives the discovery information, it requests a subscription to the alarm log.

Each alarm typically transfers 70...300 bytes of alarm status information to the subscriber. A typical discovery phase for a system of 1000 alarms (500 analog and 500 digital) takes approximately 35 seconds. This varies depending on controller loading, network loading, and message string size and languages.

During normal operation of the alarm system, the controller uses this process to send alarm data to the subscriber.

1. When an alarm event occurs, the controller time stamps the alarm data and sends it to the subscriber.

A typical alarm message size is 70 bytes for an analog alarm and 60 bytes for a digital alarm. The packet size can be as large as 500 bytes and can contain a combination of analog and digital messages.

2. The subscriber sends the alarm data to the appropriate client applications and historical databases.
3. The operator acknowledges the alarm and the acknowledge request is logged into the historical database (this time stamp is from the operator workstation).
4. The RSLinx Enterprise server sends the acknowledge request to the controller.
5. The controller receives the acknowledge request, marks the alarm as acknowledged, and time stamps the completed action back to the subscriber.
6. The controller sends the acknowledge confirmation with time stamp back to the subscriber. (This time stamp is from the controller.)
7. The subscriber sends the acknowledge to the appropriate clients and historical databases.

Because time stamps occur at multiple places during normal alarm operation, it is important to coordinate the clocks of the controllers and workstations in the system. For more information on, see Different Methods of Synchronizing Clocks with ControlLogix Controllers, Knowledgebase document 40467 at <http://www.rockwellautomation.custhelp.com>.

Alarm Log

The alarm log holds 10,000 alarm state transitions in a circular log. This replaces the alarm buffer in controllers with firmware earlier than revision 21. The original 100 KB alarm buffer is now a 2 KB subscription service with no timeouts. The alarm log is stored in extended memory.

In controllers with firmware earlier than revision 21, the controller reserves 100 KB per subscriber to buffer alarm data in case the subscriber loses its connection to the controller. Typically, this buffer holds about 1000 events.

Guideline (Firmware Earlier Than Revision 21)	Description
If the subscriber loses its connection to the controller, re-establish the connection as soon as possible.	The alarm buffer in the controller continues to buffer new alarms until either the buffer is full (100 KB) or the buffer times out. You configure the buffer timeout from 0 min...2 hr (default is 20 min) when you configure the alarm server in RSLinx Enterprise software. If the subscriber fails to reconnect by the end of this buffer time, the controller clears the buffer and reclaims the 100 KB of buffer space for normal controller operations.
You can check the status of a subscriber connection to the controller by reviewing the instruction faceplate.	The Status tab on the alarm properties in the Logix Designer application identifies whether the controller is buffering alarm data. This value is updated at the next occurrence of the alarm event.

Programmatically Access Alarm Information

Each alarm instruction has an alarm structure that stores alarm configuration and execution information. The alarm structure includes both control program elements and operator elements. The alarm instructions do not use mode settings to determine whether program access or operator access is active, so these elements are always active.

There are three ways to perform actions on an alarm instruction.

Table 9.A

Access	Alarm Structure Elements	Considerations
Control program	<ul style="list-style-type: none"> • ProgAck • ProgReset • ProgUnshelve • ProgSuppress • ProgDisable • ProgEnable 	Use controller logic to programmatically access elements of the alarming system. For example, the control program can determine whether to disable a series of alarms that are related to one root cause. Then control program could disable an alarm instruction, MyDigitalAlarm of data type DigitalAlarm, accesses a tag MyDigitalAlarm.ProgDisable.
Custom HMI	<ul style="list-style-type: none"> • OperAck • OperReset • OperShelve • OperUnshelve • OperSuppress • OperDisable • OperEnable 	Access a custom faceplate to access elements of the alarming system. For example, you can press a disable key that accesses a tag MyDigitalAlarm.OperDisable, rather than manually disable or suppress alarms individually from the alarming screens.
Standard HMI object	Not accessible	Normal operator interaction is through the alarm summary and alarm banner objects in the FactoryTalk View application. This interaction is similar to the custom HMI option, but there is no programmatic visibility or interaction.

To perform global alarm operations, access the alarm elements via the relay ladder instructions. For example, assign a BOOL tag `DisableToolA` to all the `ProgDisable` fields on the alarm relay ladder faceplates in `ToolA`. Then use the `DisableToolA` tag to disable the operation of all the alarms that use this tag.

IMPORTANT If you assign a tag to the `ProgAck`, `ProgReset`, `ProgDisable`, or `ProgEnable` functions on the alarm faceplate, do not use the alarm structure elements in the alarm data type to perform the same functions. For example, if you assign `DisableToolA` to disable an alarm `MyDigitalAlarm`, do not programmatically access `MyDigitalAlarm.ProgDisable`. Doing so can cause a condition where the faceplate requests one operation and the alarm tag requests another.

If you want to use the alarm structure elements to programmatically change the alarming system, assign those elements to a faceplate. For example, on the alarm faceplate for `ProgDisable`, assign the tag from that alarm's structure tag `MyAlarmTag.ProgDisable`. This lets you programmatically access `MyAlarmsTag.ProgDisable` in other code locations without conflict.

At the HMI and in the event log, any controller-driven events, either through the alarm structure or the alarm dialog, are logged in the historical database as Discrete Events. This includes any HMI interface that also accesses this same information via the operator elements (`.OPERxxx`). So while time stamps and events are tracked in the log, the log excludes identification of what caused the event. For example, an operator and workstation in this scenario is not tracked because they did not take action via a FactoryTalk alarm graphic object.

Shelve, Suppress, or Disable Alarms

Following ANSI/ISA-18.2-2009, Management of Alarm Systems for the Process Industries, Shelve, Suppress, and Disable are all methods to suppress indication of alarms. You can use Shelve, Suppress, and Disable functionality to track operator-initiated actions from design-initiated actions and maintenance actions. Shelve is the method to use when the operator initiates the action (equivalent to the Shelve state in ISA 18.2). Suppress is the method that the controller is expected to use to programmatically inhibit operator notification (equivalent to the Suppress-by-Design state in ISA 18.2). Disable is the method to use to inhibit the alarm for maintenance purposes (equivalent to Out-of-Service state in ISA 18.2).

- Shelve lets you clear an alarm from the alarm summary or banner while you are resolving a known alarm without continuing to view alarm information. Shelve has an automatic timeout, after which the alarm is automatically unshelved and returned to the operator view.
- Suppress lets you clear an alarm from the alarm summary or banner while you are resolving a known alarm without continuing to view alarm information. Suppress does not have an automatic timeout.
- Disable an alarm to take the alarm out-of-service in the control program.

Notes:

Optimize an Application for Use with HMI

Rockwell Automation offers these HMI (human machine interface) platforms.

Platform	Description
PanelView™ Plus terminal	Dedicated, machine-level HMI running FactoryTalk View Machine Edition software
FactoryTalk View software	Product family that consists of: <ul style="list-style-type: none"> • FactoryTalk View ME (Machine Edition) software for an open, machine-level HMI; also runs on PanelView Plus terminals • FactoryTalk View Site Edition Station software for a single-workstation, supervisory-level HMI • FactoryTalk View Site Edition distributed software for a multi-server, multi-client, supervisory-level HMI
RSView®32 software	Single-workstation or single-server, multiple-client, supervisory-level HMI

Software products that provide plant-floor device connectivity for HMI applications include:

- RSLinx Classic software, also known as RSLinx 2.x.
- RSLinx Enterprise software.

HMI Implementation Options

Method	Benefits	Considerations
Single HMI	<ul style="list-style-type: none"> • All HMI/EOI support this method • Limited number of controller connections • No server to configure and manage • Local control and monitoring 	<ul style="list-style-type: none"> • Single point of failure for visualization • Only one person can monitor one display at a time
Multiple, Independent HMI	<ul style="list-style-type: none"> • All HMI/EOI support this method • The same HMI screens can be viewed at multiple stations • Multiple people can monitor different parts of system simultaneously • Each HMI gets its own data • No central server to configure and manage • Local control and monitoring 	<ul style="list-style-type: none"> • More controller connections are required • Additional burden on controller to service all communication (program scan impact) • No sharing of data except through the controller • Adding additional HMIs has larger increase on system
Client/Server HMI	<ul style="list-style-type: none"> • The same HMI screens can be viewed at multiple stations • Server provides data to multiple clients • Fewer controller connections required • Impact on system is smaller than with multiple HMIs • Administer application at the server, not individually at the clients or multiple, independent HMIs 	<ul style="list-style-type: none"> • Server is a point of failure for all HMIs, unless you implement redundancy • Little communication overhead savings if each client wants different data • Networking knowledge that is required

Most third-party HMIs are limited to direct communication similar to the multiple HMI method.

Compare FactoryTalk View Site Edition and RView32 Software

HMI Product	Benefits	Considerations
FactoryTalk View Site Edition	<ul style="list-style-type: none"> Supports Windows 2000, Windows XP, Windows 2003, Windows Vista, and Windows 2008 operating systems Common FactoryTalk® View Studio development environment for FactoryTalk View SE and FactoryTalk View ME software (including PanelView Plus terminals) FactoryTalk enabled 	<ul style="list-style-type: none"> Does not support Windows NT operating system
RView®32	<ul style="list-style-type: none"> Supports Windows NT, Windows 2000, Windows XP, and Windows Server 2003 operating systems FactoryTalk enabled (version 7.0 and later) 	<ul style="list-style-type: none"> RView32 development environment only support RView32 software PanelBuilder® software that is used for PanelView terminals RView32 software supports only single-server architectures

Guidelines for FactoryTalk View Software

To configure a FactoryTalk View Site Edition system, a maximum of:

- Five FactoryTalk View Studio clients can have simultaneous access to an FactoryTalk View Site Edition application.
- 50 FactoryTalk View Site Edition clients can have simultaneous access to a FactoryTalk View Site Edition application.

In nonredundant applications, a maximum of:

- 10 FactoryTalk View Site Edition servers can be in a FactoryTalk View Site Edition application.
- Two FactoryTalk View Site Edition servers can be hosted on one computer.

In redundant applications, a maximum of:

- One FactoryTalk View Site Edition server can be hosted on one computer.

Contact Rockwell Software® for architectural assistance with redundant server applications or applications that require more than two FactoryTalk View Site Edition Servers and 20 FactoryTalk View Site Edition clients.

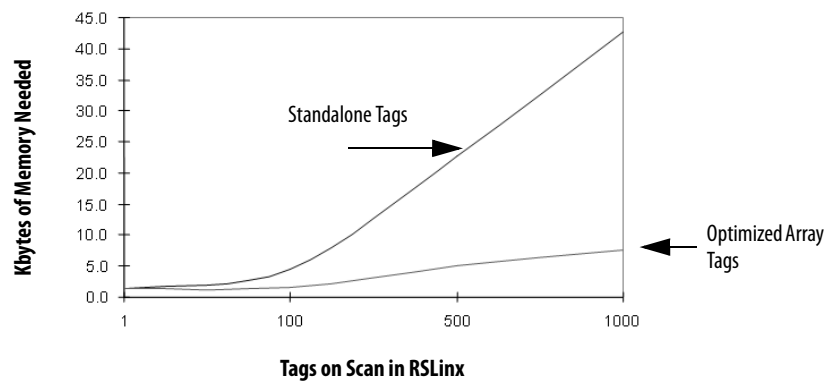
How RSLinx Software Communicates with Logix5000 Controllers

RSLinx software acts as a data server to optimize communication to HMI applications. RSLinx software groups data items into one network packet to reduce:

- The number of messages that are sent over the network.
- The number of messages a controller processes.

IMPORTANT Unless otherwise indicated, references to RSLinx software include both RSLinx Classic software and RSLinx Enterprise software.

1. When RSLinx software first connects to a Logix5000 controller, it queries the tag database and uploads definitions for all controller-scoped tags. If there are multi-layer, user-defined structures that are controller-scoped, RSLinx software just queries the upper layer.
2. When the HMI client requests data, RSLinx software queries the definitions for program-scoped tags and the lower layers of multi-layer user-defined structures.
3. RSLinx software receives requests for data items from local or remote HMI/EOI clients and combines multiple requests in optimized packets. Each data item is a simple Logix tag, array, or user-defined structure. Each optimized packet can be as large as 480 bytes of data and can contain one or more data items.
4. The Logix5000 controller allocates unused system RAM to create an optimization buffer to contain the requested data items.
 - One optimization buffer can contain as much data as can fit into one 480-byte packet (optimization is limited to 480 bytes).
 - Currently, RSLinx Enterprise software only provides optimization for array tags.
 - If you use the Logix Designer application to monitor controller RAM, you can see used memory increase.
 - The controller creates an optimization buffer for each RSLinx optimization packet in the scan.



Compare RSLinx Classic and RSLinx Enterprise Software

Comparison	RSLinx Classic (RSLinx 2.x) Software	RSLinx Enterprise Software
Supported platforms	<ul style="list-style-type: none"> Windows 2000 Windows XP Windows Server 2003 Windows Vista Business Windows Server 2008 	<ul style="list-style-type: none"> Windows CE Windows 2000 Windows XP Windows Server 2003 Windows Vista Business Windows Server 2008
Data server	<p>OPC data server</p> <p>Preferred data server for PLC/SLC platforms and applications that require complex network routings</p> <p>Maximum 10 clients per data server</p>	<p>FactoryTalk Live data server</p> <p>Preferred data server for Logix5000 platforms</p> <p>Maximum 20 clients per data server</p>
PLC/SLC systems	Maximum 20 controllers per data server via an Ethernet network	Maximum 20 controllers per data server via an Ethernet network
Logix5000 systems	<p>Maximum:</p> <ul style="list-style-type: none"> 10 controllers per data server via an Ethernet network 10,000 active (on-scan) tags per data server Three RSLinx data servers per controller 	<p>Maximum:</p> <ul style="list-style-type: none"> 20 controllers per data server via an Ethernet network 20,000 active (on-scan) tags per data server Three RSLinx Enterprise data servers per controller
User interface and event logs	Yes	<ul style="list-style-type: none"> Available user interfaces are FactoryTalk Studio software and FactoryTalk® Administration Console software Event logs are available with FactoryTalk Diagnostics software
Benefits	<ul style="list-style-type: none"> Supports topic switching with redundant ControlLogix system Supports used-defined tag optimization RSLinx Gateway software consolidates multiple HMI requests to reduce network traffic Works with an integrated OPC server 	<ul style="list-style-type: none"> Automatically handles Logix tag changes FactoryTalk® Live Data software consolidates multiple HMI requests to reduce network traffic
Considerations	<ul style="list-style-type: none"> Requires HMI to be restarted if Logix5000 controller is reloaded with changes to tags on scan Default is four connections for a read and one connection for a write 	<ul style="list-style-type: none"> Does not support topic switching with redundant ControlLogix system Optimization is limited to array tags FactoryTalk Gateway software provides OPC support Default is four connections for a read and one connection for a write

Guidelines for RSLinx Software

Guideline	Description
Use RSLinx software as the data server for multiple HMIs.	<p>For multiple HMI stations:</p> <ul style="list-style-type: none"> Leverage remote OPC (RSLinx Classic software) or FactoryTalk (RSLinx Enterprise software) software for data collection. Only the RSLinx data server is expected to have an active topic. Do not configure or use topics on the HMI stations. RSLinx software does not need to be on the HMI stations.
Do not use too many RSLinx stations.	<p>The performance of tag collection decreases as the more RSLinx stations collect data from the same controller.</p> <p>Use an RSLinx Gateway station and have the other data collection stations use remote OPC for data collection.</p>
Account for delay time when adding/removing scanned tags.	<p>When switching from one HMI screen to another, it takes time to put items in the controller on scan and take items off scan. Part of this time delay is because the controller allocates system RAM for the optimization buffer.</p> <p>To minimize this delay, when switching between HMI screens, put the items in the HMI screens on scan and leave them on scan. For example, you can create a data log to keep the items on scan. Then when switching between HMI screens, data collection continues without interruption.</p> <p>RSLinx Enterprise and FactoryTalk View Site Edition software account for this time delay. When HMI screens change, these applications deactivate tags rather than remove them from scan.</p>

Guidelines to Configure Controller Tags

Guideline	Description
Use INT data types with third-party products.	Most third-party operator interface products do not support DINT (32-bit) data types. However, there are additional performance and memory-use considerations when you use INT data types. See Guidelines for Data Types on page 54 . FactoryTalk View software supports native Logix5000 data types (including BOOL, SINT, INT, DINT, and REAL), structures, and arrays.
Group related data in arrays.	Most third-party operator interface products do not support user-defined structures. Arrays also make sure that data is in contiguous memory, which optimizes data transfer between the controller and RSLinx software or other operator interfaces. Arrays of tags transfer more quickly and take up less memory than groups of individual tags.
Use RSLinx OPC services.	Use RSLinx OPC services to bundle multiple tag requests into one message to reduce communication overhead. OPC provides better optimization than DDE.

Reference Controller Data from FactoryTalk View Software

This table shows how to reference data in a FactoryTalk View tag address.

Logix5000 Array Data Type	Description	PLC File Identifier	FactoryTalk View Tag Data Type
BOOL	Value of 0, 1, or -1	B	Digital
SINT	8-bit integer	A	Byte
INT	16-bit integer	N	Integer
DINT	32-bit integer	L	Long Integer
LINT	64-bit integer value to store date and time values	No PLC identifier	Not supported
REAL	Floating point	F	Floating Point

When addressing a Logix5000 string tag, use the address syntax `[OPC_Topic]StringTag.Data[0],SC82` to address a SINT array. The string data is stored in the SINT array `.Data` of the string tag, and you address the first element of this array (`.Data[0]`). The maximum number of characters in a STRING tag is 82. If you need more characters, then create your own user-defined structure to hold the characters.

To write data into a Logix5000 string tag from an HMI or external source, set the `.LEN` field to indicate the number of characters that are in the string. The Logix Designer application and the controller use the `.LEN` value to determine how many characters are present.

Notes:

Develop Equipment Phases

The PhaseManager™ option of the Logix Designer application gives you a state model for your equipment. It includes the following components:

- Phase to run the state model
- Equipment phase instructions for programming the phase
- PHASE data type

Guidelines for Equipment Phases

Guideline	Description
Use a separate phase for each activity of the equipment.	<p>Each phase is a specific activity that the equipment performs.</p> <ul style="list-style-type: none"> • Use one phase for standalone machines. • Make sure that each phase does an independent activity. • Keep the total number of phases and programs in a project within the limit of programs for the controller. • List the equipment that goes with each phase.
Complete one state model for each phase.	<p>Each phase runs its own set of states. A state model divides the operating cycle of the equipment into a series of states.</p> <ul style="list-style-type: none"> • Decide which state to use for the initial state after powerup. • Start with the initial state and work through the model. • Use only the states you need; skip those states that do not apply. • Use subroutines for producing and standby states. <p>The state model of an equipment phase is similar to the S88 state model. U.S. standard ISA S88.01-1995 and its IEC equivalent IEC 61512-1-1998 is commonly referred to as S88. It is a set of models, terms, and good practices for the design and operation of manufacturing systems.</p>
Separate phase code from equipment code.	<p>One advantage of a phase is that it lets you separate the procedures (recipes) for how to make the product from the control of the equipment that makes the product. This makes it easier to execute different procedures for different products by using the same equipment.</p>
Separate normal execution from exceptions.	<p>A state model makes it much easier to separate the normal execution of your equipment from any exceptions (faults, failures, off-normal conditions).</p> <ul style="list-style-type: none"> • Use a prestate routine to watch for faults. • A prestate routine is not a phase state routine. Create a routine like you do for any program and assign it as the prestate routine for the equipment phase program. • Use a state bit to limit code to a specific state. • Logix Designer application automatically makes a tag for each phase. The phase tag has bits that identify the state of the phase. For example, My_Phase.Running.
Use Equipment Phases in redundant systems.	<p>PhaseManager™ has been tested for compatibility with ControlLogix redundancy systems. See the ControlLogix Enhanced Redundancy System, firmware revision 16.81, Release Notes, publication 1756-RN650, for more information.</p>

Equipment Phase Instructions

The equipment phase instructions are available in relay ladder and structured text programming languages. You can use them in relay ladder routines, structured text routines, and SFC actions.

If you want to	Use this instruction
Signal a phase that the state routine is complete so go to the next state	Phase State Complete (PSC)
Change the state or substate of a phase	Equipment Phase Command (PCMD)
Signal a failure for a phase	Equipment Phase Failure (PFL)
Clear the failure code of a phase	Equipment Phase Clear Failure (PCLF)
Initiate communication with RSBizWare™ Batch software	Equipment Phase External Request (PXRQ)
Clear the NewInputParameters bit of a phase	Equipment Phase New Parameters (PRNP)
Create breakpoints within the logic of a phase	Equipment Phase Paused (PPD)
Take ownership of a phase to either: <ul style="list-style-type: none"> Prevent another program or RSBizWare™ Batch software from commanding a phase. Make sure another program or RSBizWare Batch software does not already own a phase. 	Attach to Equipment Phase (PATT)
Relinquish ownership of a phase	Detach from Equipment Phase (PDET)
Override a command	Equipment Phase Override (POVR)

For more information, see the PhaseManager User Manual, publication [LOGIX-UM001](#).

Manage Firmware

The Logix controllers, I/O modules, and other devices use firmware that you can update on your own. You choose the firmware revision level and decide when to update the firmware.

Guidelines to Manage Controller Firmware

Guideline	Description						
Maintain software versions and firmware revisions at the same major revision levels.	<p>At release, a specific version of software supports the features and functions in a specific revision of firmware. To use a specific revision of firmware, you must have the corresponding software version. This combination of software and firmware is considered to be compatible.</p> <p>A revision number consists of a major and minor revision number in this format xx.yy.</p> <table border="1"> <thead> <tr> <th>Where</th> <th>Is the</th> </tr> </thead> <tbody> <tr> <td>xx</td> <td>Major revision Updated every release there is a functional change</td> </tr> <tr> <td>yyy</td> <td>Minor revision Updated any time there is a change that does not affect function or interface</td> </tr> </tbody> </table>	Where	Is the	xx	Major revision Updated every release there is a functional change	yyy	Minor revision Updated any time there is a change that does not affect function or interface
Where	Is the						
xx	Major revision Updated every release there is a functional change						
yyy	Minor revision Updated any time there is a change that does not affect function or interface						
Use digitally signed firmware to maintain firmware integrity.	Some communication modules support digitally signed firmware for additional security. Once upgraded with digitally signed firmware, the module only accepts upgrade attempts that include signed firmware. The module rejects any unsigned firmware updates. To let backward compatibility occur, modules ship with unsigned firmware installed and must be upgraded to take advantage of this feature.						
Document firmware revisions.	Include software version and firmware revision information in electrical drawings and other project documentation.						
Read the associated release notes.	Always read the release notes that accompany new software versions and firmware revisions before you install them. The release notes help you to understand what has improved and changed, and also help you determine whether you need to modify your application because of the changes. In most cases, your application runs normally following an update.						
Configure modules so that the controller automatically updates firmware.	<p>Controller firmware, revision 16, includes a firmware supervisor feature that lets controllers automatically update devices. To use the firmware supervisor:</p> <ul style="list-style-type: none"> You can update Local and remote modules while in Program or Run modes, as long as their electronic keying configurations are set to Exact Match and the ControlFLASH™ Software supports the modules. Firmware kits must reside on the removable media in the controller. 						
Control that users have access to change firmware revisions.	ControlFLASH software, version 8.0 and later, is integrated with FactoryTalk Security software so you can establish update or no update privileges for users.						
Use the ControlFLASH kit manager to update only the firmware you need or have.	<p>With ControlFLASH software, version 8.0 and later, you can:</p> <ul style="list-style-type: none"> View available firmware kits before updating a device. Import and export kits to create custom kits. Delete kits as single devices or as groups by catalog number and device type. Support third-party applications to push/pull kits as needed. 						

Compare Firmware Options

Controllers ship with basic firmware that supports only updating the controller firmware to the required revision. You must update the firmware to a revision that is compatible with your version of the Logix Designer application.

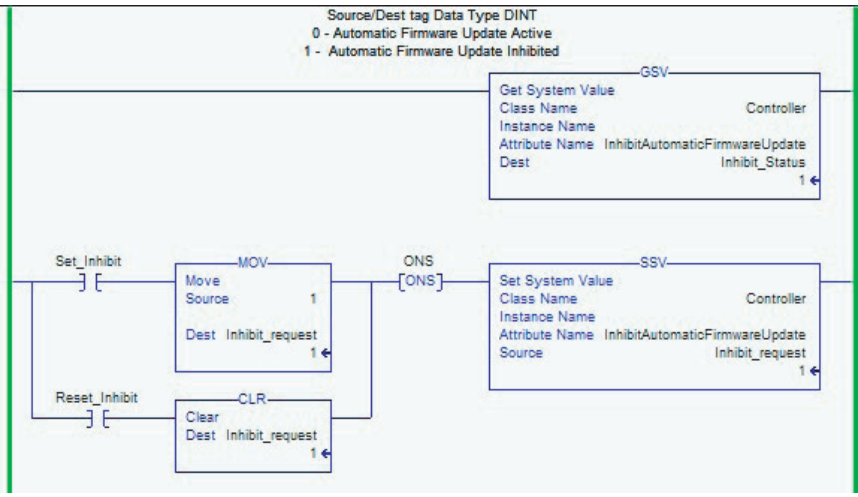
ControlFLASH Software	AutoFlash Function	Controller-based Firmware Supervisor
Standalone tool. Manually launch from desktop icon or program list.	Integrated with the Logix Designer application. The software automatically checks the controller, motion module, and SERCOS drive firmware during a project download. If the firmware is out of date or incompatible, the software prompts you to update the firmware.	Integrated on the controller removable media and run by the controller without user intervention. Controllers automatically update modules on keying mismatch situations.
Supports controllers, communication modules, I/O modules, motion modules, and newer SERCOS drives, and many other devices.	Supports the same devices as the ControlFLASH™ Software.	Supports local and remote devices that: <ul style="list-style-type: none"> • Are in the I/O tree and configured as Exact Match • Support firmware updates via the ControlFLASH Software • The hardware revision supports the firmware that is stored for that Exact Match device
Supports valid CIP path to the device to update, such as serial, DeviceNet, ControlNet, and EtherNet/IP connections.	Supports valid CIP path to the device to update, such as serial, DeviceNet, ControlNet, and EtherNet/IP connections.	Supports all communication paths to devices that reside in the controller I/O tree and that also support the ControlFLASH Software. The firmware must already be on removable media in the controller.

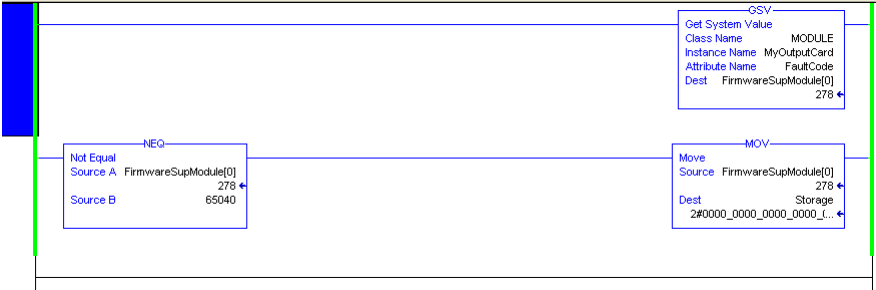
For more information, see the ControlFLASH Firmware Upgrade Software User Manual, publication [1756-UM105](#).

Guidelines for the Firmware Supervisor

The firmware supervisor feature can automatically load firmware when you replace a device in the system.

- OEMs who build multiple machines a month can have the controller update all modules and devices in the system without user intervention.
- Machines with strict regulation can require specific firmware revisions for the devices to maintain certification. The firmware supervisor helps make sure that devices are at the correct firmware revision.
- Maintenance personnel replacing failed hardware can install the replacement device and the controller automatically updates the device with the correct firmware revision.

Guideline	Description
<p>The firmware supervisor can update any Rockwell Automation device that:</p> <ul style="list-style-type: none"> • Can be placed in the I/O Configuration tree • Has electronic keying that is configured as Exact Match • Normally can be updated with ControlFLASH software 	<p>The firmware supervisor works on local I/O modules and distributed modules via EtherNet/IP, SERCOS, and ControlNet networks. On DeviceNet networks, the firmware supervisor supports local devices only, such as scanners and linking devices that reside in the I/O tree of the controller project. Because you cannot directly place a remote DeviceNet device in the I/O tree, the firmware supervisor does not manage remote DeviceNet devices.</p> <p>The firmware supervisor supports:</p> <p>Logix5000 controllers that support removable media (except for redundant controllers).</p> <p>The firmware supervisor does not manage the firmware of other standard controllers in the I/O Configuration tree.</p> <p>Safety products, including GuardLogix Safety controllers and 1791ES CompactBlock™ Guard I/O™ EtherNet/IP modules.</p> <p>The firmware supervisor does not manage the firmware of POINT Guard I/O™ modules or 1791DS CompactBlock Guard I/O DeviceNet modules.</p> <p>SERCOS drives that support updates over a SERCOS network:</p> <ul style="list-style-type: none"> • 1394 drives, firmware revision 1.85 and later. • Kinetix® 6000 drives, firmware revision 1.85 and later. • Ultra™ 3000 drives, firmware revision 1.50 and later. • 8720MC drives, firmware revision 3.85 and later. <p>Non-modular, distributed I/O products that sit directly on the network without an adapter. Distributed I/O products that require an adapter, such as POINT I/O or FLEX I/O modules, are not supported. Instead, the firmware supervisor manages the firmware for the adapters.</p> <p>The firmware supervisor does not support PanelView Plus terminals, since the terminals do not support the ControlFLASH software.</p>
<p>For the firmware supervisor to manage firmware for a device, the device must have its electronic keying that is configured for Exact Match.</p>	<p>Other modules can exist in the I/O Configuration that are not configured as Exact Match, but the firmware supervisor does not maintain the firmware for those modules.</p> <p>To disable the firmware supervisor for a specific device:</p> <p>Change the electronic keying for that device to something besides Exact Match.</p> <p>Disable firmware supervisor from either an SSV instruction or the Nonvolatile Memory tab of the controller properties.</p>
<p>Removable media must be formatted properly.</p>	<p>If you have a Secure Digital card with 4 G memory or more, format the card FAT32. If you have a Secure Digital card with less than 4 G memory, format the card FAT16.</p>
<p>Make sure that the removable media is not locked.</p>	<p>The Secure Digital card has a lock feature. The card must be unlocked to write to the card.</p>
<p>Each controller must store the firmware files for modules that are managed by the firmware supervisor on removable media.</p>	<p>Enable the firmware supervisor, from the Nonvolatile Memory tab of the controller properties. Click Load/Store. From the Automatic Firmware Updates pull-down menu, choose Store to copy it to removable media.</p> <p>The computer running the Logix Designer application must have:</p> <ul style="list-style-type: none"> • ControlFLASH Software installed. • The required firmware kits in the ControlFLASH default directory for the modules the firmware supervisor is to maintain. The Logix Designer application moves firmware kits from your computer to the removable media in the controller for the firmware supervisor to use. • Controller firmware and application logic is managed outside of firmware supervisor on the Nonvolatile Memory tab. Firmware supervisor adds to the ability to store controller firmware and logic on the removable media. If you disable the firmware supervisor, you disable the firmware supervisor updates and not the controller firmware updates that still occur when the controller image is reloaded.
<p>Enable or disable the automatic firmware updates by using GSV and SSV instructions.</p>	 <p>The diagram is a ladder logic network with the following components:</p> <ul style="list-style-type: none"> Legend: Source/Dest tag Data Type DINT; 0 - Automatic Firmware Update Active; 1 - Automatic Firmware Update Inhibited. Network 1: A normally open contact labeled 'Set_Inhibit' is connected to a 'MOV' (Move) instruction. The MOV instruction has 'Source' as 'Inhibit_request' and 'Dest' as 'Inhibit_request', both with a quantity of 1. Network 2: A normally open contact labeled 'Reset_Inhibit' is connected to a 'CLR' (Clear) instruction. The CLR instruction has 'Dest' as 'Inhibit_request' and a quantity of 1. Network 3: A normally open contact labeled 'ONS' is connected to an 'SSV' (Set System Value) instruction. The SSV instruction has 'Class Name' as 'Controller', 'Instance Name' as 'InhibitAutomaticFirmwareUpdate', and 'Attribute Name' as 'Inhibit_request', with a quantity of 1. Network 4: A normally open contact labeled 'GSV' is connected to a 'Get System Value' instruction. The GSV instruction has 'Class Name' as 'Controller', 'Instance Name' as 'InhibitAutomaticFirmwareUpdate', and 'Attribute Name' as 'Inhibit_Status', with a quantity of 1.

Guideline	Description
<p>You can monitor the status of automatic firmware updates.</p>	<p>Monitor the status of automatic firmware updates on the Nonvolatile Memory tab on the controller properties. To monitor the status of automatic firmware updates for a specific module, use GSV instructions. This example shows that the firmware supervisor encountered the wrong hardware revision for 1756-OB16D module.</p>  <pre> graph TD GSV[GSV Get System Value Class Name MODULE Instance Name MyOutputCard Attribute Name FaultCode Dest FirmwareSupModule[0] 278] --> MOV[MOV Source FirmwareSupModule[0] 278 Dest 2#0000_0000_0000_0000_I] MOV --> NEQ[NEQ Not Equal Source A FirmwareSupModule[0] 278 Source B 65040] </pre>

Access Firmware

The Logix Designer application ships with firmware update kits. Firmware revisions are also available on the Rockwell Automation website.

1. Go to <http://www.rockwellautomation.com/support/>.
2. In the left pane, under Downloads, click Firmware Updates.

The following terms and abbreviations are used throughout this manual. For definitions of terms that are not listed here, refer to the Allen-Bradley Industrial Automation Glossary, publication [AG-7.1](#).

- Add-On Instruction** An Add-On Instruction is a user-defined instruction that encapsulates executable logic and data.
- array** An array groups data of the same data type under a common name. An array tag occupies a contiguous block of memory in the controller, each element in sequence.
- atomic data type** BOOL, SINT, INT, DINT, LINT, and REAL data types.
- buffer** A temporary memory area used for queuing incoming and outgoing messages. The buffer area of a device determines how many messages can be queued for processing.
- cache** To leave a connection open for a MSG instruction that executes repeatedly.
- coarse update rate** Determines the periodic rate at which the motion task executes to compute the servo commanded position, velocity, and accelerations to be sent to the motion modules when executing motion instructions.
- compound data type** Array, structure, and string data types.
- connection** A communication link between two devices, such as between a controller and an I/O module, PanelView terminal, or another controller.
- Connections are allocations of resources that provide more reliable communication between devices than unconnected messages.
 - You indirectly determine the number of connections the controller uses by configuring the controller to communicate with other devices in the system.
- consumed tag** A tag that receives the data that is broadcast by a produced tag over an EtherNet/IP network, ControlNet network, or ControlLogix backplane. A consumed tag must be:
- Controller scope
 - Same data type (including any array dimensions) as the remote tag (produced tag).
- See [produced tag](#).
- continuous task** The continuous task runs continuously in the background. Any CPU time not allocated to other operations (such as motion, communication, and periodic tasks) is used to execute the programs within the continuous task.

controller scope Data accessible anywhere in the controller. The controller contains a collection of tags that can be referenced by the routines and alias tags in any program, as well as other aliases in the controller scope.

See [program scope](#).

direct connection A direct connection is a real-time, data transfer link between the controller and an I/O module. The controller maintains and monitors the connection with the I/O module. Any break in the connection, such as a module fault or the removal of a module while under power, sets fault bits in the data area associated with the module.

See [rack-optimized connection](#).

element An addressable unit of data that is a sub-unit of a larger unit of data. A single unit of an array or structure.

equipment phase An equipment phase is a type of program. It has routines and a set of isolated tags. It also has:

- State model
- State machine
- PHASE data type

event task An event task executes automatically based on a trigger event occurring or if a trigger event does not occur in a specific time interval.

explicit A connection that is non-time critical and is request and reply in nature. Executing a MSG instruction or executing a program upload are examples of explicit connections. Explicit refers to basic information (such as source address, data type, and destination address) that is included in every message.

See [implicit](#).

firmware revision For products that have firmware components, the product ID label identifies the firmware revision. This revision denotes the operating system for the device. The firmware revision is usually two numbers separated by a period. For example, in firmware revision 10.02, the first number (10) defines the major revision and the second number (002) defines the minor revision.

See [software version](#).

HART protocol HART (Highway Addressable Remote Transmitter) is an open protocol designed to connect analog devices.

implicit A connection that is time critical in nature. This includes I/O and produced/consumed tags. Implicit refers to information (such as source address, data type, and destination address) that is implied in the message but not contained in the message.

See [explicit](#).

- index** A reference used to specify an element within an array.
- local connection** A connection to a module in a local chassis, extended-local chassis, or any of the I/O banks configured for the controller. Communication occurs across the backplane or virtual backplane and does not require an additional communication module or adapter.
- member** An element of a structure that has its own data type and name.
- Members can be structures as well, creating nested structure data types.
 - Each member within a structure can be a different data type.
- message** A message asynchronously reads or writes a block of data to another device.
- multicast** Network technology for the delivery of information to multiple destinations simultaneously.
- network update time (NUT)** The repetitive time interval in which data can be sent on a ControlNet network. The network update time ranges from 2...100 ms.
- packet** A unit of data that is routed between an origin and a destination.
- parameter** A parameter is a value or tag passed to an instruction or returned from an instruction. An Add-On Instruction supports these parameters:
- Input (copied in)
 - Output (copied out)
 - InOut (passed by reference)
- periodic task** A periodic task executes automatically based on a preconfigured interval. This task is similar to selectable timed interrupts in PLC-5 and SLC 500 processors.
- PhaseManager option** The PhaseManager option of RSLogix 5000 software (introduced in version 15) gives you a state model for your equipment. Use the PhaseManager option to create equipment phase programs.
- postscan** A function of the controller where the logic within a program is examined before disabling the program to reset instructions and data.
- prescan** Prescan is an intermediate scan during the transition to Run mode.
- The controller performs prescan when you change from Program mode to Run mode.
 - The prescan examines all programs and instructions and initializes data based on the results.
 - Some instructions execute differently during prescan than they do during the normal scan.
- produced tag** A tag that a controller is making available for use by (consumed by) other controllers. Produced tags are always at controller scope.

See [consumed tag](#).

- product-defined data type** A structure data type that is automatically defined by the software and controller. By configuring an I/O module, you add the product-defined data type for that module.
- program** A set of related routines and tags. Each program contains program tags, a main executable routine, other routines, and an optional fault routine.
- program scope** Data accessible only within the current program. Each program contains a collection of tags that can only be referenced by the routines and alias tags in that program.
- See [controller scope](#).
- rack-optimized connection** For digital I/O modules, you can select rack-optimized communication. A rack-optimized connection consolidates connection usage between the controller and all the digital I/O modules in the chassis (or DIN rail). Rather than having individual, direct connections for each I/O module, there is one connection for the entire chassis (or DIN rail).
- See direct connection.
- remote connection** A connection to a module in a remote chassis or DIN rail. Communication requires a communication module and/or adapter.
- requested packet interval (RPI)** When communicating over a the network, this is the maximum amount of time between subsequent production of input data.
- Typically, this interval is configured in microseconds.
 - The actual production of data is constrained to the largest multiple of the network update time that is smaller than the selected RPI.
- routine** A set of logic instructions in a single programming language, such as a ladder diagram. Routines provide the executable code for the project in a controller (similar to a program file in a PLC or SLC controller).
- scheduled connection** A scheduled connection is unique to ControlNet communication. A scheduled connection lets you send and receive data repeatedly at a predetermined rate that is the requested packet interval (RPI). For example, a connection to an I/O module is a scheduled connection because you repeatedly receive data from the module at a specified rate. Other scheduled connections include connections to the following:
- Communication devices
 - Produced/consumed tags

On a ControlNet network, you must use RSNetWorx for ControlNet software to enable all scheduled connections and establish a network update time (NUT).

- software version** The product ID label of a software products identifies the software version. This version denotes the functional version of the software. The software version is usually two numbers separated by a period. For example, in software version 10.02, the first number (10) defines the major revision and the second number (02) defines the minor revision.
- See [firmware revision](#).
- state machine** A state machine does the following:
- Calls the main routine (state routine) for an acting state.
 - Manages the transitions between states with minimal coding.
 - Makes sure that the equipment goes from state to state along a valid path.
- state model** A state model divides the operating cycle of your equipment into a series of states. Each state is an instant in the operation of the equipment. It's the actions or conditions of the equipment at a given time.
- structure** Some data types are a structure.
- A structure stores a group of data, each of which can be a different data type.
 - Within a structure, each individual data type is called a member.
 - Like tags, members have a name and data type.
 - You create your own user-defined structure by using any combination of individual tags and most other structures.
 - To copy data to a structure, use the COP instruction.
- system overhead timeslice** Specifies the percentage of controller time (excluding the time for periodic tasks) that is devoted to communication and background functions (system overhead).
- tag** A named area of controller memory where data is stored. Tags are the basic mechanism for allocating memory, referencing data from logic, and monitoring data.
- task** A scheduling mechanism for executing a program. By default, each new project file contains a preconfigured continuous task. You configure additional periodic and event tasks, as needed.
- unconnected message** An unconnected message is a message that does not require connection resources. An unconnected message is sent as a single request/response.
- unicast** Network technology for the delivery of information to a single destination.

user-defined data type (UDT) A UDT is a data structure you define. A user-defined data type groups different types of data into a single named entity. You define the members of the user-defined data type. Like tags, the members have a name and data type.

virtual communication relationship (VCR) A VCR is a channel that provides for the transfer of data between FOUNDATION Fieldbus devices. The number of VCRs required to send data or receive data depends on the device and type of data. The type of VCR determines whether the transfer is scheduled or unscheduled.

- A client/server VCR is for queued, unscheduled, user-initiated, and one-to-one communication.
- A report distribution VCR is for queued, unscheduled, user-initiated, and one-to-many communication.
- A publisher/subscriber VCR is for buffered, one-to-many communication.

A

access
 firmware 122
 module object 33

Access the Module Object 33

add-on instruction
 guidelines 48
 postscan logic 37
 prescan 37

addresses
 serial bit 60

alarm
 and events
 FactoryTalk 103
 lbuffer 108
 log 108
 process 107
 shelve, suppress, or disable 109

alias tags
 creating 63

applications
 HMI 111

array
 guidelines 56
 index
 guidelines 57
 indirect addresses 56
 tag storage 55

atomic data types 53

B

base tag
 guidelines 62

bit tags 59

block-transfer messages
 guidelines 101

buffer
 alarm 108
 message storage 98
 routine 36

C

cache
 messages 98

CIP Sync 20

code reuse
 guidelines 41

communication
 module connections 18
 MSG instruction 97
 RSLinx data packets 113

comparison

HMI software 112
 import/export, add-on instructions 50
 program parameters, add-on instructions 52
 programming languages 34
 scheduled and unscheduled ControlNet 94
 subroutines, add-on instructions 49

compound data types 53**configuration**

Logix-based alarms 105
 tags 62

connection

communication module 18
 controller 17

considerations

periodic, event tasks 30
 task 27

consumed tag

event task 69

continuous

task 26
 lowest priority 25
 task configuration 29

controller

connection 17
 dual-core 14
 memory estimation 16
 mode switch 21
 resources 14
 RSLinx
 software memory 16
 -scoped tags 113
 tag guidelines 115
 task execution 23

ControlNet network

guidelines 92
 scheduled and unscheduled comparison 94
 topology 91

creating

alias tags 63

D**data**

scope guidelines 64
 type guidelines 54

DeviceNet network

guidelines 95
 topology 94

disable

alarms 109

dual-core

controller 14

E**equipment phases** 117

guidelines 117
 instructions 118

estimate

- controller memory 16

EtherNet/IP network

- guidelines 89
- switches 90
- topology 88

event

- task 26
 - configuration 30
 - considerations 30
 - consumed tag 69
 - guidelines 30

executable code

- routines 24

execution

- project 28
- timer 38

F**FactoryTalk**

- alarms and events 103
- software guidelines 112

firmware

- access 122
- management 119
- options 120
- supervisor guidelines 120

G**guidelines**

- block-transfer messages 101
- controller firmware 119
- controller tags 115
- ControlNet network 92
- DeviceNet network 95
- equipment phases 117
- EtherNet/IP network 89
- FactoryTalk View software 112
- firmware supervisor 120
- Logix-based alarm instructions 103
- messages 100
- RSLinx software 114

H**HMI**

- optimization 111

I**indexed routine 35****inline duplication 35****instructions**

- equipment phases 118

L**log**

- alarm 108

logic

- routine application code 34

Logix5000 controller

- resources 14

Logix-based

- alarm
 - configuration 105
 - instruction guidelines 103

M**manage**

- firmware updates 119
- system overhead 32

map tags 101**memory**

- estimation 16
- RSLinx software estimation 16

message

- block-transfer guidelines 101
- cache 98
- guidelines 100
- storage buffer 98

mode switch

- controller 21

module object 33

- path attribute 33

MSG

- communication 97

N**network**

- ControlNet guidelines 92
- ControlNet topology 91
- DeviceNet guidelines 95
- DeviceNet topology 94
- EtherNet/IP guidelines 89
- EtherNet/IP switches 90
- EtherNet/IP topology 88
- guidelines 87
- services 87
- unscheduled and scheduled ControlNet 94
- unscheduled ControlNet guidelines 93

P**packet**

- RSLinx data 113

path attribute 33**periodic**

- task 26
 - configuration 29
 - considerations 30

phases

- equipment 117
- PhaseManager option 117

postscan

- add-on instruction 37
- SFC logic 37

prescan

- add-on instruction 37

priority level

task 25

produced and consumed

RPI 68

tag guidelines 67

tags 67

program

considerations 24

languages comparison 34

methods 35

routines, tags 23

-scoped tags 113

project

execution 28

R**resources**

Logix5000 controllers 14

routine

considerations 24

executable code 24

programming logic 34

programs 23

RPI

produced and consumed tags 68

RSLinX

classic and enterprise software 114

network data packet 113

software

controller memory estimate 16

guidelines 114

RSLogix 5000 software

PhaseManager option 117

S**serial bit addresses** 60**services**

network 87

SFC

logic postscan 37

online editing 39

shelve

alarms 109

storage

message buffer 98

Stratix

switches 91

string data types

guidelines 61

suppress

alarms 109

switch

controller mode 21

switches

EtherNet/IP network 90

Stratix 91

synchronization

time 20

system overhead

manage timeslice 32

timeslice 31

T**table**

mapping 101

tag

configuration 62

controller-scoped 113

descriptions 66

maps 101

name guidelines 64

produced and consumed 67

program-scoped 113

task

configure controller execution 23

considerations 24, 27

continuous, periodic, event 26

priority level 25

types 25

time

synchronization 20

timer execution 38**timeslice**

manage system overhead 32

system overhead 31

topology

ControlNet network 91

DeviceNet network 94

EtherNet/IP network 88

U**UDT**

guidelines 58

unscheduled ControlNet

network guidelines 93

updating

firmware 119

Notes:

Rockwell Automation Support

Rockwell Automation provides technical information on the Web to assist you in using its products.

At <http://www.rockwellautomation.com/support> you can find technical and application notes, sample code, and links to software service packs. You can also visit our Support Center at <https://rockwellautomation.custhelp.com/> for software updates, support chats and forums, technical information, FAQs, and to sign up for product notification updates.

In addition, we offer multiple support programs for installation, configuration, and troubleshooting. For more information, contact your local distributor or Rockwell Automation representative, or visit <http://www.rockwellautomation.com/services/online-phone>.

Installation Assistance

If you experience a problem within the first 24 hours of installation, review the information that is contained in this manual. You can contact Customer Support for initial help in getting your product up and running.

United States or Canada	1.440.646.3434
Outside United States or Canada	Use the Worldwide Locator at http://www.rockwellautomation.com/rockwellautomation/support/overview.page , or contact your local Rockwell Automation representative.

New Product Satisfaction Return

Rockwell Automation tests all of its products to help ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs to be returned, follow these procedures.

United States	Contact your distributor. You must provide a Customer Support case number (call the phone number above to obtain one) to your distributor to complete the return process.
Outside United States	Please contact your local Rockwell Automation representative for the return procedure.

Documentation Feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete this form, publication [RA-DU002](#), available at <http://www.rockwellautomation.com/literature/>.

Rockwell Automation maintains current product environmental information on its website at <http://www.rockwellautomation.com/rockwellautomation/about-us/sustainability-ethics/product-environmental-compliance.page>.

Rockwell Otomasyon Ticaret A.Ş., Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1756-RM094I-EN-P - September 2015

Supersedes Publication 1756-RM094H-EN-P - November 2012

Copyright © 2015 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.